HOMEWORK#4 QUESTİON 1

MUSTAFA TOKGÖZ

171044077

İ) A + ((B – C * D) / E) + F – G / H

To convert Postfix

Next Token	Action	Stack	Postfix
A	Append A to Postfix		A
+	Stack is empty and push + to stack	+	A
(	(opening parantes is pushed to stack	+(	A
(	(opening parantes is pushed to stack	+((	A
B	Append B to postfix	+((	AB
-	- minus is pushed to stack	+((-	AB
C	Append C to postfix	+((-	ABC
*	It pushes * to stack	+((-*	ABC
)	Pop stack until (one by one	+(	ABC*-
/	it pushes / to stack	+(/	ABC*-
E	Append E to postfix	+(/	ABC*-E
)	Pop stack until (one by one	+	ABC*-E/
+	+ is equal precedence to the top one So first pop from the stack to postfix then it pushes +	+	ABC*-E/+
F	Append F to postfix	+	ABC*-E/+F
-	- is equal precedence to the top one So first pop from the stack to postfix then it pushes - to stack	-	ABC*-E/+F+
G	Append G to postfix	-	ABC*-E/+F+G
/	It pushes / to stack	-/	ABC*-E/+F+G
H	Append H to postfic	-/	ABC*-E/+F+GH
End of input	Stack is not empty So it pop s one by one		ABC*-E/+F+GH/-

Postfix = ABC*-E/+F+GH/-

To find prefix

First , We take reverse the expression then convert postfix , at the end we take reverse again.

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

Next Token	Action	Stack	Postfix
H	Append H to Postfix		H
/	Stack is empty and push / to stack	/	H
G	Append G to Postfix	/	HG
-	- minus is lower presendece than / So It pops and push -	-	HG/
F	Append F to postfix	-	HG/F
+	+ is equal precedence to - So it pops then push +	+	HG/F-
(	(opening parantes is pushed to stack	+(	HG/F-
E	Append E to postfix	+(	HG/F-E
/	It pushes / to stack	+(/	HG/F-E
(	(opening parantes is pushed to stack	+(/(	HG/F-E
D	Append D to postfix	+(/(	HG/F-ED
*	It pushes * to stack	+(/(*	HG/F-ED
C	Append C to postfix	+(/(*	HG/F-EDC
-	- is lower precedence than * So it pops and push - to stack	+(/(-	HG/F-EDC*
B	Append B to postfix	+(/(-	HG/F-EDC*B
)	Pop stack until (one by one	+(/	HG/F-EDC*B-
)	Pop stack until (one by one	+	HG/F-EDC*B-/

+	+ is equal precedence to the top one So first pop from the stack to postfix then it pushes +	+	HG/F-EDC*B+
A	Appends A to postfix	+	HG/F-EDC*B+A
End of input	Stack is not empty So it pop s one by one		HG/F-EDC*B+A+

To find Prefix , We take reverse of expression that is HG/F-EDC*B+A+.

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

Evaluating Part

Postfix :

Expression	Action	Stack
ABCD*-E/+F+GH/-	Push A	A
ABCD*-E/+F+GH/-	Push B	A B
ABCD*-E/+F+GH/-	Push C	A B C
ABCD*-E/+F+GH/-	Push D	A B C D
ABCD*-E/+F+GH/-	Pop C and D then Push C*D	A B C*D
ABCD*-E/+F+GH/-	Pop B and C*D and push B-C*D	A B-C*D
ABCD*-E/+F+GH/-	Push E	A B-C*D E
ABCD*-E/+F+GH/-	Pop B-(C*D) and E then push B-(C*D)/E	A (B-C*D)/E
ABCD*-E/+F+GH/-	Pop (B-C*D)/E and A then push A+((B-C*D)/E)	A+((B-C*D)/E)
ABCD*-E/+F+GH/-	Push F	A+((B-C*D)/E) F

ABCD*-E/+F+GH/-	Pop A+((B-C*D)/E) and F then push A+((B-C*D)/E)+ F	A+((B-C*D)/E)+ F
ABCD*-E/+F+GH/-	Push G	A+((B-C*D)/E)+ F G
ABCD*-E/+F+GH/-	Push H	A+((B-C*D)/E)+ F G H
ABCD*-E/+F+GH/-	Pop G and H then push G/H	A+((B-C*D)/E)+ F G/H
ABCD*-E/+F+GH/-	Pop A+((B-C*D)/E)+ F and G/H then push A+((B-C*D)/E)+ F - G/H	A+((B-C*D)/E)+ F-G/H
ABCD*-E/+F+GH/- _	Pop A+((B-C*D)/E)+ F-G/H and Stack is empty then result is A+((B-C*D)/E)+ F-G/H	

Expression=ABCD*-E/+F+GH/-

Result = A+((B-C*D)/E)+ F-G/H

Prefix :

Expression	Action	Stack
+A+/-B*CDE-F/GH	Push H	H
+A+/-B*CDE-F/GH	Push G	H G
+A+/-B*CDE-F/GH	Pop G and H then push G/H	G/H
+A+/-B*CDE-F/GH	Push F	G/H F
+A+/-B*CDE-F/GH	Pop F and G/H then push F-G/H	F-G/H
+A+/-B*CDE-F/GH	Push E	F-G/H E
+A+/-B*CDE-F/GH	Push D	F-G/H E D
+A+/-B*CDE-F/GH	Push C	F-G/H E D C

+A+/-B*CDE-F/GH	Pop C and D then push C*D	F-G/H E C*D
+A+/-B*CDE-F/GH	Push B	F-G/H E C*D B
+A+/-B*CDE-F/GH	Pop B and C*D then push B-C*D	F-G/H E B-C*D
+A+/-B*CDE-F/GH	Pop B-C*D and E then push (B-C*D)/E	F-G/H (B-C*D)/E
+A+/-B*CDE-F/GH	Pop (B-C*D)/E and F-G/H then push (B-C*D)/E + F-G/H	(B-C*D)/E+F-G/H
+A+/-B*CDE-F/GH	Push A	(B-C*D)/E+F-G/H A
+A+/-B*CDE-F/GH	Pop A and (B-C*D)/E+F-G/H then push A+(B-C*D)/E+F-G/H	A+(B-C*D)/E+F-G/H
_+A+/-B*CDE-F/GH	Pop A+(B-C*D)/E+F-G/H and Stack is empty then result is A+(B-C*D)/E+F-G/H	

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

Result = A+(B-C*D)/E+F-G/H

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

Postfix :

Next Token	Action	Stack	Postfix
!	Stack is empty ,It pushes ! to stack	!	
(	It pushes (to stack	!(	
A	Append A to Postfix	!(	A
&&	It pushes && to stack	!(&&	A

!	It pushes ! to stack becouse ! has higher precedence than &&	!(&&!	A
(	It pushes (to stack	!(&&!(	A
(	It pushes (to stack	!(&&!((	A
B	Append B to postfix	!(&&!((	AB
<	It pushes < to the stack	!(&&!((<	AB
C	Append C to postfix	!(&&!((<	ABC
)	Pop stack until (one by one	!(&&!(	ABC<
||	It pushes || to stack	!(&&!(||	ABC<
(	It pushes (to stack	!(&&!(||(	ABC<
C	Append C to postfix	!(&&!(||(	ABC<C
>	It pushes > to stack	!(&&!(||(>	ABC<C
D	Append D to postfix	!(&&!(||(>	ABC<CD
)	Pop stack until (one by one	!(&&!(||	ABC<CD>
)	Pop stack until (one by one	!(&&!	ABC<CD>||
)	Pop stack until (one by one	!	ABC<CD>||!&&
||	|| is lower precedence than ! so it pops then pushes ||	||	ABC<CD>||!&&!
(	It pushes (to stack	||(	ABC<CD>||!&&!
C	Append C to postfix	||(	ABC<CD>||!&&!C
<	It pushes < to stack	||(<	ABC<CD>||!&&!C
E	Append E to postfix	||(<	ABC<CD>||!&&!CE
)	Pop stack until (one by one	||	ABC<CD>||!&&!CE<
End of input	Stack is not empty So it pop s one by one		ABC<CD>||!&&!CE<||

Postfix : ABC<CD>||!&&!CE<||

Prefix :

To find prefix

First , We take reverse the expression then convert postfix , at the end we take reverse again.

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

Next Token	Action	Stack	Postfix
(	It pushes (to stack	(	
E	Append E to postfix	(	E
<	It pushes < to the stack	(<	E
C	Append C to postfix	(<	EC
)	Pop stack until (one by one		EC<
||	It pushes || to stack	||	EC<
(	It pushes (to stack	||(	EC<
(	It pushes (to stack	||((	EC<
(	It pushes (to stack	||(((	EC<
D	Append D to postfix	||(((	EC<D
>	It pushes > to the stack	||(((>	EC<D
C	Append C to postfix	||(((>	EC<DC
)	Pop stack until (one by one	||((	EC<DC>
||	It pushes || to stack	||((||	EC<DC>
(	It pushes (to stack	||((||(	EC<DC>
C	Append C to postfix	||((||(	EC<DC>C
<	It pushes < to stack	||((||(<	EC<DC>C
B	Append B to postfix	||((||(<	EC<DC>CB
)	Pop stack until (one by one	||((||	EC<DC>CB<

)	Pop stack until (one by one	||(	EC<DC>CB<||
!	It pushes ! to stack	||(!	EC<DC>CB<||
&&	&& has lower precedence than ! so it pops then it pushes &&	||(&&	EC<DC>CB<||!
A	Append A to postfix	||(&&	EC<DC>CB<||!A
)	Pop stack until (one by one	||	EC<DC>CB<||!A&&
!	It pushes ! to stack	||!	EC<DC>CB<||!A&&
End of input	Stack is not empty So it pop s one by one		EC<DC>CB<||!A&&!||

To find Prefix , We take reverse of expression that is EC<DC>CB<||!A&&!||

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

Evaluating Part

Postfix : ABC<CD>||!&&!CE<||

Expression	Action	Stack
ABC<CD>||!&&!CE<||	Push A	A
ABC<CD>||!&&!CE<||	Push B	A B
ABC<CD>||!&&!CE<||	Push C	A B C
ABC<CD>||!&&!CE<||	Pop B and C Then push B<C	A B<C
ABC<CD>||!&&!CE<||	Push C	A B<C C
ABC<CD>||!&&!CE<||	Push D	A B<C C D

ABC<CD>||!&&!CE<||	Pop C and D Then push C>D	A B<C C>D
ABC<CD>||!&&!CE<||	Pop B<C and C>D then push B<C||C>D	A (B<C)||(C>D)
ABC<CD>||!&&!CE<||	Pop (B<C)||(C>D) then push !((B<C)||(C>D))	A !((B<C)||(C>D))
ABC<CD>||!&&!CE<||	Pop A and !((B<C)||(C>D)) then push A&&!((B<C)||(C>D))	A&&!((B<C)||(C>D))
ABC<CD>||!&&!CE<||	Pop A&&!((B<C)||(C>D)) then push !(A&&!((B<C)||(C>D)))	!(A&&!((B<C)||(C>D)))
ABC<CD>||!&&!CE<||	Push C	!(A&&!((B<C)||(C>D))) C
ABC<CD>||!&&!CE<||	Push E	!(A&&!((B<C)||(C>D))) C E
ABC<CD>||!&&!CE<||	Pop C and E then push C<E	!(A&&!((B<C)||(C>D))) C<E
ABC<CD>||!&&!CE<||	Pop !(A&&!((B<C)||(C>D))) And C<E then push !(A&&!((B<C)||(C>D))) ||C<E	!(A&&!((B<C)||(C>D))) ||C<E
ABC<CD>||!&&!CE<||_	Pop !(A&&!((B<C)||(C>D))) ||C<E then stackis empty So result is !(A&&!((B<C)||(C>D))) ||(C<E)	

Expression = ABC<CD>||!&&!CE<||

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

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

Expression	Action	Stack
||!&&A!||<BC>CD<CE	Push E	E
||!&&A!||<BC>CD<CE	Push C	E C
||!&&A!||<BC>CD<CE	Pop C and E then push C<E	C<E
||!&&A!||<BC>CD<CE	Push D	C<E D
||!&&A!||<BC>CD<CE	Push C	C<E D C
||!&&A!||<BC>CD<CE	Pop C and D then push C>D	C<E C>D
||!&&A!||<BC>CD<CE	Push C	C<E C>D C
||!&&A!||<BC>CD<CE	Push B	C<E C>D C B
||!&&A!||<BC>CD<CE	Pop B and C then push B<C	C<E C>D B<C
||!&&A!||<BC>CD<CE	Pop B<C and C>D then push (B<C)||(C>D)	C<E (B<C)||(C>D)
||!&&A!||<BC>CD<CE	Pop (B<C)||(C>D) then push !((B<C)||(C>D))	C<E !((B<C)||(C>D))
||!&&A!||<BC>CD<CE	Push A	C<E !((B<C)||(C>D)) A
||!&&A!||<BC>CD<CE	Pop A and !((B<C)||(C>D)) then push A&&!((B<C)||(C>D))	C<E A&&!((B<C)||(C>D))

||!&&A!||<BC>CD<CE	Pop A&&!((B<C)||(C>D)) then push !(A&&!((B<C)||(C>D)))	C<E !(A&&!((B<C)||(C>D)))
||!&&A!||<BC>CD<CE	Pop !(A&&!((B<C)||(C>D))) and C<E Then push !(A&&!((B<C)||(C>D))) ||C<E	!(A&&!((B<C)||(C>D))) ||(C<E)
_||!&&A!||<BC>CD<CE	Pop then Stack is empty So result is !(A&&!((B<C)||(C>D))) ||(C<E)	

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

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