CSE 222 HW4

Melihcan Çilek
1801042092

13 April 2020

Question 1:

i) A+((B–C*D)/E)+F–G/H

Assume that
A = 1, B = 2, C = 3, D = 4,
E = 5, F = 6, G = 7, H = 8
	1 + ((2 – 3 * 4) / 5) + 6 – 7/8	1 + ((2 – **3 * 4**) / 5) + 6 – 7/8
		1 + ((**2 – 12**) / 5) + 6 – 7/8
		1 + (**(-10) / 5**) + 6 – 7/8
		1 + (**-2 + 6**) – 7/8
		1 + 4 – 7/8
		1 + **4 – 0.875**
		1 + 3.125
		4.125

Expression	Stack	Postfix(String)	Explanation
A	(	A	First character came to the postfix
+	(+	A	Because of '+' is a symbol not a letter, we push it to our stack and because of priority of '(' less than '+', we push it to our stack
(	(+(	A	If '(' comes, we push it to stack so we push '(' character stack again
B	(+((	A B	B is a letter so we append it end of the postfix
-	(+((-	A B	Character '-'s priority is higher than '(' character which is peek of the stack so we push '-' to the stack
C	(+((-	A B C	C is a letter so we append it end of the postfix
*	(+((-*	A B C	Character '*'s priority is higher than '-' character which is peek of the stack so we push '*' to the stack
D	(+((-*	A B C D	D is a letter so we append it end of the postfix
)	(+(	A B C D *-	Because ')' character came, we pop all symbols to end of the postfix till we see '(' character.
/	(+(/	A B C D *-	Character '/'s priority is higher than '(' character which is peek of the stack so we push '/' to the stack
E	(+(/	A B C D *- E	E is a letter so we append it end of the postfix
)	(+	A B C D *- E /	Because ')' character came, we pop all symbols to end of the postfix till we see '(' character.
+	(++	A B C D *- E /	Priority of '+' which is peek of the stack and '+' which is next character's priorities are the same so we can push it to stack.
F	(++	A B C D *- E / F	F is a letter so we append it end of the postfix
-	(-	A B C D *- E / F + +	'-'s priority is lower than plus because while calculating, order of '-' operation is important so we pop both '+' operations from stack to end of the postfix
G	(-	A B C D *- E / F + + G	G is a letter so we append it end of the postfix
/	(-/	A B C D *- E / F + + G	Character '/'s priority is higher than '-' character which is peek of the stack so we push '/' to the stack
H	(-/	A B C D *- E / F + + G H	H is a letter so we append it end of the postfix
)		A B C D *- E / F + + G H / -	Because ')' character came, we pop all symbols to end of the postfix till we see '(' character.Our expression is over and our stack is empty so our conversion is successful.

Question 1 Postfix Expression Evaluation:

Assume that
A = 1, B = 2, C = 3, D = 4,
E = 5, F = 6, G = 7, H = 8
	1 2 3 4 * - 5 / 6 + + 7 8 / -	1 2 **3 4 *** - 5 / 6 + + 7 8 / -
		1 **2 12 -** 5 / 6 + + 7 8 / -
		1 **(-10) 5** / 6 + + 7 8 / -
		1 **(-2) 6** + + 7 8 / -
		1 4 + 7 8 / -
		5 7 8 / -
		5 0.875 -
		4.125

Expression	Stack	Prefix(String)	Explanation
H	(	H	First character came to the prefix, '(' and ')' characters added beginning and end of the string so '(' added to stack
/	(/	H	Because of '/' is a symbol not a letter, we push it to our stack and because of priority of '(' less than '/', we push it to our stack
G	(/	H G	G is a letter so we append it end of the prefix
-	(-	H G /	Because of '-' is a symbol not a letter and priority of '-' less than '/', '/' will be added to prefix and '-' character pushed to stack
F	(-	H G / F	F is a letter so we append it end of the prefix
+	(-+	H G / F E	E is a letter so we append it end of the prefix
/	(-+(/	H G / F E	Character '/'s priority is higher than '(' character which is peek of the stack so we push '/' to the stack
(	(-+(/(	H G / F E	We push '(' character into stack anyway
D	(-+(/(	H G / F E D	D is a letter so we append it end of the prefix
*	(-+(/(*	H G / F E D	Character '*'s priority is higher than '(' character which is peek of the stack so we push '*' to the stack
C	(-+(/(*	H G / F E D C	C is a letter so we append it end of the prefix
-	(-+(/(-	H G / F E D C *	Because of '-' is a symbol not a letter and priority of '-' less than '*', '*' will be added to prefix and '-' character pushed to stack
B	(-+(/(-	H G / F E D C * B	B is a letter so we append it end of the prefix
)	(-+(/	H G / F E D C * B -	Because ')' character came, we pop all symbols to end of the prefix till we see '(' character
)	(-+	H G / F E D C * B -/	Because ')' character came, we pop all symbols to end of the prefix till we see '(' character.
+	(-++	H G / F E D C * B -/	Priority of '+' which is peek of the stack and '+' which is next character's priorities are the same so we can push it to stack.
A	(-++	H G / F E D C * B -/ A	A is a letter so we append it end of the prefix
)		H G / F E D C * B - / A + + -	Because ')' character came, we pop all symbols to end of the prefix till we see '(' character.Our expression is over and our stack is empty so our conversion is successful.
		- + + A / - B * C D E F / G H	At the end, we will reverse this string shown as Prefix column

Question 1 Prefix Expression Evaluation:

Assume that A = 1, B = 2, C = 3, D = 4, E = 5, F = 6, G = 7, H = 8	- + + 1 / - 2 * 3 4 5 6 / 7 8	- + + 1 / - 2 * 3 4 5 6 **7 / 8**
		- + + 1 / - 2 * **3 4** 5 6 **0.875**
		- + + 1 / **- 2 12** 5 6 0.875
		- + + 1 / **10 5** 6 0.875
		- + **+ 1 2** 6 0.875
		- **+ 3 6** 0.875
		- 9 0.875
		8.125

Question 2:

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

Assume that
A = 1, B = 0, C = 1,
D = 0, E = 1
	! (1 &&! ((0 < 1)||(1 > 0)))||(1 < 1)	! (1 &&! **((0 < 1)**||**(1 > 0))**)||(1 < 1)
		! (1 &&! **(1||1)**))||(1 < 1)
		! (1 && **!1**))||(1 < 1)
		! **(1 &&0)**)||(1 < 1)
		!0||(1 < 1)
		1||**(1 < 1)**
		1||0
		1

Expression	Stack	Postfix(String)	Explanation
!	!		First character came to the stack because '!' character is a symbol
(	!(		'(' always added to stack until ')' symbol
A	!(	A	A is a letter so we append it end of the postfix. 'A' is added as first character
&&	!(&&	A	'&&' symbol's priority is more than '(' symbol so it will be added to stack
!	!(&&!	A	'!' symbol's priority is more than '&&' symbol so it will be added to stack
(	!(&&!(	A	'(' always added to stack until ')' symbol
(	!(&&!((	A	'(' always added to stack until ')' symbol
B	!(&&!((	AB	B is a letter so we append it end of the postfix.
<	!(&&!((<	AB	'<' symbol's priority is more than '(' symbol so it will be added to stack
C	!(&&!((<	ABC	C is a letter so we append it end of the postfix.
)	!(&&!	ABC<	Because ')' character come, we pop all the characters until we peek '(' character at the top. After than we pop '(' character at the top
||	!(&&!||	ABC<	'||' symbol's priority is more than '!' which is top of the stack symbol so it will be added to stack
(	!(&&!||(	ABC<	'(' always added to stack until ')' symbol
C	!(&&!||(	ABC<C	C is a letter so we append it end of the postfix.
>	!(&&!||(>	ABC<C	'>' symbol's priority is more than '(' symbol so it will be added to stack
D	!(&&!||(>	ABC<CD	D is a letter so we append it end of the postfix.
)	!(&&!||	ABC<CD>	Because ')' character come, we pop all the characters until we peek '(' character at the top. After than we pop '(' character at the top
)	!	ABC<CD>||!&&	Because ')' character come, we pop all the characters until we peek '(' character at the top. After than we pop '(' character at the top
)		ABC<CD>||!&&!	Because ')' character come, we pop all the characters until we peek '(' character at the top. After than we pop '(' character at the top
||	||	ABC<CD>||!&&!	There is no element at the stack so '||' character will be pushed to stack
(	||(	ABC<CD>||!&&!	'(' always added to stack until ')' symbol
C	||(	ABC<CD>||!&&!C	C is a letter so we append it end of the postfix.
<	||(<	ABC<CD>||!&&!C	'<' symbol's priority is more than '(' which is top of the stack symbol so it will be added to stack
E	||(<	ABC<CD>||!&&!CE	E is a letter so we append it end of the postfix.

Question 2 Postfix Expression Evaluation:

Assume that A = 1, B = 0, C = 1, D = 0,E = 1	101<10>||!&&!11<||	1**01<**10>||!&&!11<|| 1**110>**||!&&!11<|| 1**11||**!&&!11<|| **11!**&&!11<|| **10&&**!11<|| **0!**11<|| **111**<|| **10**|| **1**

Expression	Stack	Prefix(String)	Explanation
)	)		’)’ always added to stack until ’(’ symbol
E	)	E	E is a letter so we append it end of the prefix.
<	)<		’<’ symbol’s priority is more than ’)’ symbol so it will be added to stack
C	)<	EC	C is a letter so we append it end of the prefix
(		EC<	Because ’(’ character come, we pop all the characters until we peek ’)’ character at the top. After than we pop ’)’ character at the top
||	||	EC<	First character came to the stack because ’||’ character is a symbol
)	||)	EC<	’)’ always added to stack until ’(’ symbol
)	||))	EC<	’)’ always added to stack until ’(’ symbol
)	||)))	EC<	’)’ always added to stack until ’(’ symbol
D	||)))	EC<D	D is a letter so we append it end of the prefix.
>	||)))>	EC<D	’>’ symbol’s priority is more than ’)’ symbol so it will be added to stack
C	||)))>	EC<DC	C is a letter so we append it end of the prefix.
(	||))	EC<DC>	Because ’)’ character came, we pop all symbols to end of the prefix till we see ’(’ character then append them end of the prefix string.
||	||))||	EC<DC>	’||’ symbol’s priority is more than ’)’ symbol so it will be added to stack
)	||))||)	EC<DC>	’)’ always added to stack until ’(’ symbol
C	||))||)	EC<DC>C	C is a letter so we append it end of the prefix.
<	||))||)<	EC<DC>C	’<’ symbol’s priority is more than ’)’ symbol so it will be added to stack
B	||))||)<	EC<DC>CB	B is a letter so we append it end of the prefix.
(	||))||	EC<DC>CB<	Because ’)’ character came, we pop all symbols to end of the prefix till we see ’(’ character then append them end of the prefix string.
(	||)	EC<DC>CB<||	Because ’)’ character came, we pop all symbols to end of the prefix till we see ’(’ character then append them end of the prefix string.
!	||)!	EC<DC>CB<||!	’!’ symbol’s priority is more than ’)’ symbol so it will be added to stack
&&	||)!&&	EC<DC>CB<||!	’&&’ symbol’s priority is less than ’!’ symbol so ’!’ will be appended to prefix string, ’&&’ will be pushed to stack
A	||)!&&	EC<DC>CB<||!A	A is a letter so we append it end of the prefix.
(	||	EC<DC>CB<||!A&&!	Because ’)’ character came, we pop all symbols to end of the prefix till we see ’(’ character then append them end of the prefix string.
!	||!	EC<DC>CB<||!A&&!	’!’ symbol’s priority is more than ’||’ symbol so it will be added to stack
		EC<DC>CB<||!A&&!||	We pop all symbols at the stack and append all the symbols that are poped from stack we append to postfix
		||!&&A!||<BC>DC<CE	At the end, we will reverse this string shown as Prefix column

Question 2 Prefix Expression Evaluation:

Assume that A = 1, B = 0, C = 1, D = 0,E = 1	||!&&1!||<01>01<11	||!&&1!||<01>01**<11** ||!&&1!||<01**>010** ||!&&1!||**<010**0 ||!&&1!||**10**0 ||!&&1**!1**0 ||!**&&10**0 ||**!0**0 **||10** **1**