Exercise #(2)

SWE2015-41 2021 Fall Semester

Student ID: 2021311013

Name: 최효승 / Choi Hyoseung

I have neither given nor received help from others in this exercise. (최효승)

I consent that my solution can be used for the class. (Agree / Disagree) -> Agree

0. Any comment to Prof. or TA (leave blank if you have no comment)

1. Analysis of Problem (up to 0.5 page)

This problem is to convert Infix to Postfix. Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands. Postfix expression: The expression of the form a b op. **When an operator is followed for every pair of operands**.

For example, (2+1)*3 -> 21+3*

(2*1+2)*2 -> 21*2+2*

But I have to concern about parenthesis. If there is some parenthesis, I have to calculate first than any operators.

And this problem can be solved by using Stack(Data Structure).

2. Explain your solution (up to 0.5 page)

I get operators and operands as string. And with For iteration, I can access each characters in string. If character is operand(number), then just print number.

If character is operator, then push it into stack. However before pushing, I have to check the elements priority in the stack. If the priority of the incoming operator is lower than or equal to the top, pop and print the top until this process finishes.

*와 /는 priority를 5로 잡고, +와 -는 1로 우선도를 잡습니다. 특히 주의해야 하는 부분은 괄호의 경우입니다. 여는 괄호의 경우는 operator Stack에 push합니다. It is eliminated when closing operator comes. When closing operator ')' comes, I will POP from Stack until I can access opening operator '('.

At the end of the infix notation, pop all operators.

3. Analysis of the results of your solution (up to 0.5 page)

If infix is (3+8)*2+(4-1)/3

Print	Stack
	(
3	(
3	(+
3 8	(+

And then I meet Closing operator ')', so I will pop all operators until I meet opening operator '('

But I don't print opening element

3 8 +	
3 8 +	*
3 8 + 2	*

And then I meet ‘+’ operator, but its priority is lower than element already exist in Stack which is ‘*’. So I have to pop first, and then push

3 8 + 2 *	+
3 8 + 2 *	* (
3 8 + 2 * 4	* (
3 8 + 2 * 4	* (-
3 8 + 2 * 4 1	* (-
3 8 + 2 *. 4 1 -	*
3 8 + 2 * 4 1 -	* /

At the end, pop all operators.

3 8 + 2 * 4 1 – 3 / *