HW8: infix evaluation, postfix to infix conversion

제출은 gitLab을 통해서 하며 **http://hconnect.hanyang.ac.kr/2017\_CSE2010\_수업번호/2017\_CSE2010\_수업번호\_학번.git 에 HW8라는 폴더를 만들어 진행.**

\* 프로그램 제출간 유의사항

- 소스코드에는 주석이 있어야 함.

- 주어진 구조체와 input.txt를 사용해야 함

**- 숙제의 소스코드 평가는 linux ubuntu 16.04.2 LTS 버전 gcc 5.4.0에서 함.**

\* 보고서 제출간 유의사항

- 작성한 소스 코드가 첨부되어야 하며, 실행결과가 첨부되어야 함.

- 분량은 제한이 없으나 1~2apge로 간략하게 설명.

- 보고서는 hw8\_학번.확장자(doc, docx, pdf)로 제출.

**제출시간: '17.5.31(23:59) 까지**

**\*지연제출**

- 24시간 이내는 해당 과제 50% 감점, 48시간 이내는 75% 감점.

- 지연제출자는 E-mail(casualab@hanyang.ac.kr)과 gitlab에 모두 제출.

- E-mail제목: "hw8\_학번\_자신의 수업 요일(수, 목)\_이름 " 형식으로 제출.

This assignment has two topics. 'Topic1' is to evaluate the expression written in infix, and 'Topic2' is postfix to infix conversion. You have to implement both topics using the given code(HW8\_given.c).

\* Topic1:  
 First, you have to read a infix expression from the given input1 file(input1.txt). Second, convert a postfix expression from the infix expression using stack ADT. Last, evaluate the value of the postfix expression.

\* Topic2:

This topic uses the given input2 file(input2.txt). Implement a function to convert from a postfix expression to infix expression.

1. Input

The infix expression is in the input1 file(input1.txt) that will be used in the topic1. And the postfix expression is in the input2 file(input2.txt) that will be use in the topic2. A detailed specification of operators and operands is provided below.

Input1.txt

|  |
| --- |
| (2+3)\*2-4 |

Input1 Result



Input2.txt

|  |
| --- |
| 35+4\* |

Input2 Result



* Available operators: +, -, \*, /
* Operands: single-digit numbers (1, 2, 3, 4, 5, 6, 7, 8, and 9)
* Conditions:
* The expression should be no more than 20 characters.
* No exception handling is required for checking whether the input file exists.

2. Structure and function descriptions

#define MAX\_STACK\_SIZE 20

typedef int element;

typedef struct {

element stack[MAX\_STACK\_SIZE];

int top;

}StackType;

void infix\_to\_postfix(char infix[],char postfix[])

- Converts the infix expression to postfix(Parameters and return types can be changed at your situation).

char\* postfix\_to\_infix(char exp [])

- Converts the postfix expression to infix(Parameters and return types can be changed at your situation).

int prec(char op)

- Return operator priority

.

int postfixEval(char exp[])

- Calculate the postfix expression.

\* The following functions are functions used in the stack, so they are not discussed in this homework.

void init(StackType \*s)

int is\_empty(StackType \*s)

int is\_full(StackType \*s)

void push(StackType \*s, element item)

element pop(StackType \*s)

element peek(StackType \*s)

3. Postfix evaluation algorithm

* Iteratively obtain tokens until you meet the end of an expression.
* There are two rules for popping and pushing the operands from/to the stack:
* When you meet an operand (number), push it onto the stack.
* When you meet an operator, pop two operands from the stack and perform the operation, and push the result back to the stack.

4. Postfix to infix conversion algorithm

* Scan the Postfix String from Left to Right.
* If the character is an Operand, then Push it on to the Stack.
* If the character is an Operator, then Pop Operator 1 and Operand 2 and concatenate them using Infix notation where the Operator is in between the Two Operands.
* The resultant expression is then pushed on the Stack
* Repeat the above steps till the Postfix string is not scanned completely.

5. Program description

* name : hw8\_학번.c
* input : a list of operations in a file (an input file name is given as a command line argument. See the example in “1. input” on the first page)
* output : the corresponding result in the standard output