Lab 4: Infix to postfix conversion using Stack

In this week, we will develop a program that converts infix to postfix. Your program reads an infix expression from the given input file, and translates it to the postfix expression by using Stack ADT. Your Stack ADT should have five essential functions such as *pop*, *push*, *top*, *createStack*, and *removeStack*. You can finish the entire program at home. But, you should finish making Stack ADT during the lab session and confirm it with TA before you leave the lab. After you implement Stack ADT, you can test Stack ADT in your main function during the lab.

1. Input

Obtain an infix expression from the given input file (expr_input.txt). The expression ends with #. A detail specification of operators & operands is provided below.

```
thkim@ubuntu:/data/thkim/dslab$ gcc -o p4 p4.c
thkim@ubuntu:/data/thkim/dslab$ ./p4 lab4_input.txt
original infix form : 4*(7+3%6)-(4/2)+9-(2*3)#
converted postfix form : 4736%+*42/-9+23*-#
```

- Available operators: +, -, *, /, and %
- Operands: single-digit numbers (1,2,3,4,5,6,7,8, and 9)
- Conditions:
    - The expression should be no more than 100 characters.
    - Parentheses are allowed.
    - The delimiter for the end of the expression is '#'.
    - No exception handling is required for checking whether the input file exists.
- Operator precedence:

token	precedence
()	3
* / %	2
+ -	1

2. Data structure

```
struct Stack {
    char *key;
```

```
    int top;
    int max_stack_size;
};
```

3. Algorithm for translating infix to postfix expression

- The translation will be conducted by scanning the infix expression.
- There are several rules for popping and pushing the operators from/to the stack.
  - When you meet an operand, print it.
  - When you meet an operator, push it as long as the precedence of the operator at the top of the stack is less than the precedence of the incoming operator.
  - When you meet an operator whose precedence is equal to or less than the precedence of the top of the stack, pop the top element and print it.
  - When you meet the right parenthesis, pop all the operators until we reach the corresponding left parenthesis.
  - When you reach the end of expression, pop all the operators from the stack.
  - For detailed instructions on handling parenthesis, please refer to the textbook
- See more details in the textbook.

4. Program description

- name : p4.c
- input : an infix expression in a file
- output : postfix expression on the standard output
- main function:

Submit to the course website (https://portal.hanyang.ac.kr) your source code and a written report. Your report should include the description of your own implementation.