Programming Part

Arithmetic Expression (20 pts)

Problem Description

Given an infix arithmetic expression that contains non-negative integers, binary operators $+, -, *, /, \%$, and parentheses, convert it to the postfix expression by

```
INFIX-TO-POSTFIX(infix)
 1  S = empty stack
 2  for each token in infix
 3      if token is a number
 4          output token
 5      elseif token is an operator
 6          if token is '('
 7              PUSH(S, token)
 8          elseif token is ')'
 9              while NOT(IS-EMPTY(S)) and PEEP(S) is not '('
10                  output POP(S)
11              POP(S) // which is '('
12          else
13              while NOT(IS-EMPTY(S)) and PEEP(S) ≥ token in terms of precedence
14                  output POP(S)
15              PUSH(S, token)
16  while NOT(IS-EMPTY(S))
17      output POP(S)
```

Then, evaluate the value of the postfix expression by

POSTFIX-EVAL($postfix$)
1 S = empty stack
2 **for** each *token* in $postfix$
3 **if** *token* is a number
4 PUSH($S, token$)
5 **else //** *token* is an operator
6 $n2$ = POP(S)
7 $n1$ = POP(S)
8 PUSH(S, CALCULATE($n1, token, n2$))
9 **return** POP(S)

Note that $/, \%$ follows the *integer arithmetic* in C and evaluates to integer values.

Input

Each line contains a valid infix expression without any spaces.

Output

For each infix expression in the input, output a line of

`[postfix]=[value]`

where `[postfix]` is the postfix expression without spaces, and `[value]` is the final value.

Constraint

- There are always 3 expressions (i.e. 3 lines) in each testcase.
- The length of each expression will not exceed 4000.
- Each given number is a non-negative integer that holds within `unsigned int`.
- Within the operation of the POSTFIX-EVAL algorithm, any number will not exceed the range of `long long`.
- When running / or %, there will not be any division by zero.

Sample Testcases

Sample Input 1

```
8+9
8+9*6
8+9*6-4
```

Sample Output 1

```
89+=17
896*+=62
896*+4-=58
```

Sample Input 2

```
(8+9)*6
(8+9)*(6-4)
(8-9)%(6-4)
```

Sample Output 2

```
89+6*=102
89+64-*=34
89-64-%=-1
```

Sample Input 3

```
(8+9)/6%4
(8-9)/6%4
(8-9-6)%4
```

Sample Output 3

```
89+6/4%=2
89-6/4%=0
89-6-4%=-3
```

Hint

- By design, you can pass this homework by simulating the two algorithms properly. There is no need for other arithmetic calculations or cuts.