

**자료 구조 과제 3**

**과목명 자료구조**

**담당교수 김희철교수님**

**제출일 20210408**

**전공 컴퓨터전자시스템**

**학번 201904458**

**이름 이준용**

* **문제 1-1 번**

한 줄로 입력된 괄호들 (,),{,},[,]에 대하여, 모든 괄호가 짝이 맞는 경우 1을 출력하고, 그렇지 않으면 0을 출력한다.

|  |  |
| --- | --- |
| 입력 예 1 | 출력 예 1 |
| ()[{a}] | 1 |

|  |  |
| --- | --- |
| 입력 예 2 | 출력 예 2 |
| ()[{a}] {()]() | 0 |

|  |  |
| --- | --- |
| 입력 예 3 | 출력 예 3 |
| ()[{a}]) {()]() | 0 |

|  |  |
| --- | --- |
| 입력 예 4 | 출력 예 4 |
| [()]{[{a}] {()}() | 0 |

* **알고리즘 및 자료구조**

**-**solution 함수 : d 변수 -> 괄호 쌍

사용자에게 입력 받은 문자열을 solution 함수에서 검사함.

열린 괄호를 만났을 경우 push

닫힌 괄호를 만났을 경우 pop

쌍이 아닐 경우 0 출력

쌍일 경우 1 출력

* **느낌점**

괄호 검사 문제를 코딩하면서 자료구조인 스택에 대해서 어떤 원리로 작동하는 모델인지 확실히 알게 되었습니다.

* **프로그램 코드**

# 1.제목:한국외대 자료구조 과제(괄호 검사)  
# 2.날짜:20210404  
def solution(s):  
 d = {  
 ')' : '(',  
 '}' : '{',  
 ']' : '['  
 }  
 stack = []  
 for c in s: # input으로 문자열을 받아서 한 문자씩 루프  
 if c in '({[': # 한 문자가 열린괄호 일경우  
 stack.append(c) # 스택 리스트에 append  
 elif c in ')}]': # 한 문자가 닫힌괄호 일경우  
 if stack:  
 top = stack.pop() # top 변수에 스택에 저장되어있는 제일 위의 문자 pop  
 if d[c] != top: # d 변수의 set에서 괄호의 짝이 아닐 경우  
 return False # False 반환  
 else:  
 return False # 위의 모든 경우 아닐 경우 반환  
 return len(stack) == 0 # 스택을 비움  
  
p = input()  
if(solution(p) == False): # solution 함수에서 반환한 값이 False 일 경우  
 print("0") # 0출력  
else:  
 print("1") # 반환 값이 True 일 경우 1출력

* **문제 1-2 번**

한 줄로 입력된 괄호들 (,),{,},[,]에 대하여, 모든 괄호가 짝이 맞는 경우 1을 출력하고, 그렇지 않으면 괄호검사 알고리즘에 의하여 **처음으로 발견된 (짝을 찾지 못한 괄호) 오류에 대하여, 입력 문자열에서 이 괄호의 위치(인덱스)를 출력한 후 다음 중 하나를 출력한다**.

)에 대응하는 (가 없을 경우: error1

}에 대응하는 {가 없을 경우: error2

]에 대응하는 [가 없을 경우: error3

(에 대응하는 )가 없을 경우: error4

{에 대응하는 }가 없을 경우: error5

[에 대응하는 ]가 없을 경우: error6

|  |  |
| --- | --- |
| 입력 예 1 | 출력 예 1 |
| ()[{a}] | 1 |

|  |  |
| --- | --- |
| 입력 예 2 | 출력 예 2 |
| ()[{a}] {()]() | 12 error3 |

|  |  |
| --- | --- |
| 입력 예 3 | 출력 예 3 |
| ()[{a}]) {()]() | 7 error1 |

|  |  |
| --- | --- |
| 입력 예 4 | 출력 예 4 |
| [()]{[{a}] {()}() | 4 error5 |

* **알고리즘 및 자료구조**

1-1번 문제에서 solution함수 구조에서 오류(456)을 검출하는 부분을 추가함

Stack에 원소가 남아있을 경우

사용자에 입력 받은 스트링을 거꾸로 뒤집어서 스택에서 pop한 원소가 비교하고

오류 인덱스와 error 반환

메인 함수에는 error1 부터 6까지 짝을 찾지 못한 경우에 해당하는 괄호를 검사해서 그에 해당하는 오류를 출력함.

* **느낌점**

테스트 케이스에서 절반만 맞았습니다. Error 456 부분에서 오류인 괄호의 자리를 번호를 어떻게 반환해야 될지 정확히 해결하지 못한 부분이라고 생각합니다.

* **프로그램 코드**

# 1.제목:한국외대 자료구조 과제(괄호 검사)  
# 2.날짜:20210405  
def solution(s): # 괄호 검사 함수  
 d = { # d 라는 변수에 짝이 맞는 괄호 set으로 만듦  
 ')': '(',  
 '}': '{',  
 ']': '[',  
 '{': '}',  
 '(': ')',  
 '[': ']'  
 }  
 stack = [] # stack변수  
 count = 0 # 오류난 곳 인데스 반환 변수  
 for c in s: # 한 문자씩 for 루프  
 if c in '({[': # 한 문자가 열린 괄호 일 경우  
 count += 1 # 한 문자를 검사 했으므로 count +1  
 stack.append(c) # 열린 괄호는 스택에 append  
 elif c in ')}]': # 한 문자가 닫힌 괄호 일 경우  
 if stack:  
 top = stack.pop() # top 변수에 스택에서 맨위의 원소 pop  
 if d[c] != top: # 괄호가 짝이 아닐 경우  
 errnum = count # 오류 난 인덱스 반환 변수 errnum  
 err.append(d[c]) # 전역 변수 err 리스트에 대응하는 괄호가 없는 괄호는 append  
 return False, errnum # 에러일 경우 False와 에러 인덱스 반환  
 else:  
 errnum = count # 닫힌괄호가 아닌 대응하는 열린 괄호가 없는 경우  
 err.append(d[c]) # 전역 변수 err에 닫힌 괄호가 없는 괄호를 append  
 return False, errnum # 에러일 경우 False와 에러 인덱스 반환  
 elif c == ' ': # 문자가 빈칸일 경우 자리수 +1  
 count += 1  
 continue # 빈칸은 오류 아님 자리인데스 + 1  
 else:  
 count += 1 # 괄호가 아닌 다른 숫자나 문자가 올경우 그냥 자리 인덱스 +1  
 if (len(stack)!=0): # 스택에 값이 남아 있을경우 ( 오류 검출 )  
 top = stack.pop() # top변수에 처음으로 발견된(짝을 찾지 못한 괄호) pop  
 count = len(s) # count변수에 입력받은 문자열 길이  
 s = ''.join(reversed(s)) # 입력받은 문자열을 뒤집음  
 for c in s: # 뒤집힌 문자열에서 처음부터 검사해서 처음으로 발견된 자리 인덱스  
 count -= 1 # 검출하는 알고리즘  
 if c in top:  
 errnum = count  
 err.append(d[c]) # 오류난 괄호의 짝을 err리스트에 append  
 return False, errnum # 오류 검출 했으므로 False와 오류난 자리 인덱스 반환  
 elif c == ' ':  
 count += 1  
 continue  
  
 return True, 1, len(stack) == 0 # 오류 검출 함수에서 정상적으로 입력 받았을 경우 True 반환  
  
err = [] # 짝을 찾지 못한 경우 그 괋호에 대응 하는 괄호 반환 변수  
count = 0 # 오류 자리 인덱스 반환 변수  
p = input()  
x = solution(p)  
if str(solution(p)[0]) == "False": # 짝이 맞지 않는 경우  
 if err[:-1] == ['(']: # [:-1] 인것은 처음으로 발견된 ( 짝을 찾지 못한 경우) 오류  
 print(x[1], "error1") # 를 반환 하기 위함  
 elif err[:-1] == ['{']:  
 print(x[1], "error2")  
 elif err[:-1] == ['[']:  
 print(x[1], "error3")  
 elif err[:-1] == [')']:  
 print(x[1], "error4")  
 elif err[:-1] == ['}']:  
 print(x[1], "error5")  
 elif err[:-1] == [']']:  
 print(x[1], "error6")  
else:  
 print("1") # 짝이 맞는 경우 1출력

* **문제 2-1 번**

후위 (postfix) 표기 수식을 읽어 그 결과값을 출력하는 프로그램을 작성하시오. **모든 피연산자는 정수 상수이고, 연산자는 이항연산자인 ‘+’, ‘-’, ‘\*’, ‘//’, ‘%‘이다. 여기서,** //는 몫 연산자이다. 식의 마지막은 ;으로 끝난다. **피연산자와 연산자 사이에는 공백이 있다.**

제약조건: 스택을 사용한다.

입력: 후위 표기 수식이 한 줄에 주어진다.

출력: 후위 표기 수식의 결과값을 출력한다. **단, 후위 표기 수식에 오류가 있을 경우 error를 출력한다.**

입력 예 1: 출력 예 1:

5 4 – 6 + 3 \* ; 21

입력 예 2: 출력 예 2:

5 15 – 20 \* 3 // ; -66

입력 예 3: 출력 예 3:

30 -10 - \* 10 ; 400

**입력 예 4: 출력 예 4:**

**30 -10 - + 10 40 \* ;**  **error**

* **알고리즘 및 자료구조**

Class Stack

-Push 함수

-pop 함수(스택에 pop기능과 스택이 비어 있을 경우 indexError예외 처리함)

-evaluation함수 : 사용자로부터 입력 받은 후위 수식을 계산하는 함수

스택에 원소가 남아있을 경우 error 이므로 오류 출력 하는 부분 추가함.

-Main 함수 : 사용자로부터 입력 받은 문자열 공백으로 spilt 해서 evaluation함수에 넘겨줌

후위 수식 값 출력

* **느낌점**

후위 수식과 중위 수식 전위 수식에 대해 완벽히 이해 할 수 있는 시간 이었습니다. 특히 스택 모델을 사용하여 후위 수식을 계산하는 함수부분을 코딩하며 spilt 함수와 pop push 기능을 알게 되었습니다. 클래스를 만들고 메인 함수에서 클래스와 함수를 이용하여 문제를 해결해 나아가는 메커니즘을 잘 활용할 수 있게 되었습니다. 어떤 오류가 발생하는지 해당 오류에 대해서 예외 처리를 어떻게 하는지에 대해 확실하게 알아갈 수 있는 시간이었습니다.

* **프로그램 코드**
* # 1.제목:한국외대 자료구조 과제(후위 수식 계산)  
  # 2.날짜:20210406  
  class Stack:  
   def \_\_init\_\_(self): # 스택 생성자 초기화  
   self.items = []  
    
   def push(self, j): # push 함수를 쓸경우 스택에 해당인자 push  
   self.items.append(j)  
    
   def pop(self): # pop에서 스택이 비어 있을 경우 indexError예외 처리하고 error 반환  
   try:  
   return self.items.pop()  
   except IndexError:  
   print('error')  
    
  def evaluation(expr): # expr : list expr은 공백을 제외한 원소들로만 구성되어있음  
   s = Stack()  
   op = '+-\*//%' # op = ('+','-','\*','/')  
   for item in expr: # 리스트의 원소를 하나씩 검사  
   if item in op: # 원소가 연사자일 경우 pop 두번해서 연산하고 다시 push 하는 과정  
   right\_opr = s.pop() # 첫번째 원소 pop 두 번째 원소 pop  
   left\_opr = s.pop() # operand: 피연산자, operator: 연산자  
   if item == '+':  
   s.push(left\_opr + right\_opr)  
   elif item == '-':  
   s.push(left\_opr - right\_opr)  
   elif item == '\*':  
   s.push(left\_opr \* right\_opr)  
   elif item == '//':  
   s.push(left\_opr // right\_opr)  
   elif item == '%':  
   s.push(left\_opr % right\_opr)  
   elif item == ';': # 원소가 세미콜론일 경우 for구문 break  
   break  
   else:  
   s.push(int(item)) # 원소가 연산자가 아닐 경우는 이문제에서는 정수 와 상수뿐이므로 push  
   if len(s.items) > 1: # 스택에 원소가 남아 있을 경우  
   return 'error' # 오류임  
   else:  
   return s.pop() # 오류가 아닐경우 후위수식 계산한 값 pop해서 반환  
    
  def main():  
   expr = input().split() # expr: ['20','30','-','15','\*',';'] 문자열로 되어 있음  
   try:  
   print(evaluation(expr))  
   except TypeError: # 타입 에러일 경우 pop한 피 연산자들의 타입이 int형이 아닐 경우  
   print('') # 오류 except  
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   main()
* **문제 2-2 번**

(2) (1)의 확장

**연산자 전후에 공백이 없을 수도 있다. 단, 빼기 연산자 – 다음에 피연산자(정수)가 나올 경우 공백으로 분리된다.**

예를 들어, 10 + 20 70 30-+, 10 20- 70 30-+와 같이 입력될 수 있다. (문법에 맞지 않을 경우) 연산자에 대한 두 피연산자가 없을 경우 error1을 출력하고 피연산자 수가 많을 경우 error2를 출력한다. 예를 들어 10 20 - - 50이나 10 30 50 + 등은 문법에 맞지 않는 경우이다.

|  |  |
| --- | --- |
| 입력 예 1 | 출력 예 1 |
| 10 20 70 30-++; | 70 |

|  |  |
| --- | --- |
| 입력 예 2 | 출력 예 2 |
| 10 20 - - 50 ; | error1 |

|  |  |
| --- | --- |
| 입력 예 3 | 출력 예 3 |
| 10 30 50 +; | error2 |

* **알고리즘 및 자료구조**

2-1번에서 사용한 함수 그대로 사용하였고

- Switch 함수 -> 사용자의 입력을 공백으로 spilt해서 받아온 리스트를 정수 따로 연산자 따로 변환 하여 새로운 리스트에 추가하는 함수

-Switch 함수: 스택 모델을 사용

받아온 문자열 원소가 길이가 1인경우와 1보다 큰 경우로 나눠서 append하는 부분 두가지

길이가 1인경우 -> append

1보다 큰 경우 -> 1. 연산자를 만났을 경우

2. 숫자를 만났을 경우

3. /을 만났을 경우(//은 문자열 길이가 2이므로 따로 처리함)

-main함수

입력 받은 문자열 -> switch함수 -> evaluation -> 결과값 출력

* **느낌점**

교수님께서 수요일 5시반에 피드백 받은 내용을 위주로 다시 논리를 처음부터 잘 세워서 함수구조를 새로 정의했습니다. // 처리와 - - 정수 부분을 처리하는 과정에서 컴파일러가 문자열을 어떻게 처리하는지에 대한 메커니즘을 디버깅을 통해서 알 수 있었습니다. 함수에서 생각한 논리대로 구조를 어떻게 하면 조금 더 간단하게 코드를 작성할 수 있을 지에 대해 고민을 많이 하게 되는 시간이었습니다.

테스트 케이스를 절반 밖에 맞추지 못하였는데 오류인지 아닌지 판단하는 부분에서 해결하지 못했습니다.

* **프로그램 코드**

# 1.제목:한국외대 자료구조 과제(후위 수식 계산)  
# 2.날짜:20210407  
class Stack:  
 def \_\_init\_\_(self): # 스택 생성자 초기화  
 self.items = []  
  
 def push(self, j): # push 함수를 쓸경우 스택에 해당인자 push  
 self.items.append(j)  
  
 def pop(self): # pop에서 스택이 비어 있을 경우 indexError예외 처리하고 error 반환  
 try:  
 return self.items.pop()  
 except IndexError:  
 print('error1')  
  
def evaluation(expr): # expr : list expr은 공백을 제외한 원소들로만 구성되어있음  
 s = Stack()  
 op = '+-\*//%' # op = ('+','-','\*','//', '%')  
 for item in expr: # 리스트의 원소를 하나씩 검사  
 if item in op: # 원소가 연사자일 경우 pop 두번해서 연산하고 다시 push 하는 과정  
 right\_opr = s.pop() # 첫번째 원소 pop 두 번째 원소 pop  
 left\_opr = s.pop() # operand: 피연산자, operator: 연산자  
 if item == '+':  
 s.push(left\_opr + right\_opr)  
 elif item == '-':  
 s.push(left\_opr - right\_opr)  
 elif item == '\*':  
 s.push(left\_opr \* right\_opr)  
 elif item == '//':  
 s.push(left\_opr // right\_opr)  
 elif item == '%':  
 s.push(left\_opr % right\_opr)  
 elif item == ';': # 원소가 세미콜론일 경우 for구문 break  
 break  
 else:  
 s.push(int(item)) # 원소가 연산자가 아닐 경우는 이문제에서는 정수 와 상수뿐이므로 push  
 if len(s.items) > 1: # 스택에 원소가 남아 있을 경우  
 return 'error2' # 오류임  
 else:  
 return s.pop() # 오류가 아닐경우 후위수식 결과값 pop해서 반환  
  
  
def switch(exprStr): # 사용자의 입력을 공백으로 spilt해서 받아온 리스트를 정수 따로 연산자따로 변환해서 리스트화 하는 함수  
 x = []  
 op = '+', '\*', '%', '-', ';' # - 연산자는 - 정수부분 처리를 위해 따로 처리  
 oi = '0123456789' # // 연산자도 문자길이가 2이므로 따로 처리  
 val = '' # //와 - 연산자를 제외한 자머지 연사자를 처리하기 위한 변수  
 val1 = '' # // 연사자를 처리하기 위한 변수  
 for c in exprStr: # 리스트의 원소 한개씩 검사  
 if len(c) == 1: # 원소가 길이 한개 인 것들은 append함  
 x.append(c)  
 elif c[0] in '-': # -(피연산자) 0보다 작은 정수의 경우는 스트링 [0]검사해서 '-'있으면 append  
 x.append(c)  
 elif len(c) > 1: # 피연산자와 연산자가 붙어 있는 경우 떼어내는 부분  
 for s in c: # 한 글자씩 검사  
 if s in op: # 한글자가 연산자일 경우  
 x.append(val) # val변수는 연산자 이전에 숫자가 있을 경우 val에 저장하여  
 val = '' # 연산자를 만날경우 숫자 반환하고 다시 초기화  
 x.append(s) # 새로운 리스트 x에 append 하는 부분  
 elif s in oi: # oi는 숫자임  
 val += s # 숫자일 경우 val 변수에 string으로 이어 붙이기  
 elif s == '/': # // 연산자도 길이가 2이므로 문자 한개씩 검사하 하다가 //만날경우  
 val1 += '/' # 숫자 string 이어 붙이기와 같은 메커니즘으로 //가 될경우 append  
 if val1 == '//':  
 x.append(val1) # 새로운 리스트 x에 append합니다.  
 x.append(val) # 그냥 길이가 1이상인 숫자만 있을 경우 val에 저장된 숫자 append  
 val = ''  
 while '' in x:  
 x.remove('') # x리스트 안에 ['']원소는 모두 삭제  
 return x  
  
  
def main():  
 expr = input().split()  
 result = switch(expr)  
 try:  
 print(evaluation(result))  
 except: # 모든 예외는 그냥 '' 출력하게 함.  
 print('')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

* **문제 2-3 번**

다음 명령어를 수행하는 프로그램을 작성하시오. 단 아래의 명령어에 주어지는 수식은 후위 표기 수식이고, 연산자들은 +, -, \*, //, %이고, 피연산자는 정수형 상수 혹은 변수이다. 수식의 연산자 전후, 피연산자 전후에 공백이 있다(첫 번째 피연산자 전에는 공백이 없을 수도 있다).

**명령어 종류 및 양식:**

명령어는 다음의 세 가지 종류가 있고, 각 명령어는 ;으로 끝난다.

**(i) set 명령어**

set 명령어 양식: “set 변수 수식 ;”

여기서, 변수는 알파벳으로 시작하는 문자열이고, 수식은 후위 표기 수식임.

set 명령어 수행 결과: 수식을 계산하여 그 결과값을 변수가 참조.

**(ii) eval 명령어**

eval 명령어 양식: “eval 수식 ;”

여기서, 수식은 후위 표기 수식임.

eval 명령어 수행 결과: 수식을 계산하여 그 결과값을 출력.

**(iii) quit 명령어**

quit 명령어 양식: “quit ;”

quit 명령어 수행 결과: 프로그램을 종료함.

**제약조건: postfix 수식 계산을 위한 스택을 사용하여야 함**

**힌트: 변수와 값을 저장하는 딕셔너리를 이용**

명령어 수행 예 출력

eval 3 5 + ; 8 # 후위 표기 수식 결과값 8을 출력

eval -3 5 + ; 2 # 후위 표기 수식 결과값 2를 출력

eval 3 5 8 - + \* ; error1 # error1을 출력: postfix 수식 오류

eval 9 34 + 3 - ; 40 # 후위 표기 수식 결과값 40을 출력

eval 13 ; 13 # 후위 표기 수식 결과값 13을 출력

set a 60 20 + ; # 출력결과는 없고, 변수 a가 80을 참조

set b 15 ; # 출력결과는 없고, 변수 b가 15를 참조

eval a b + 50 - ; 45 # 후위 표기 수식의 결과값 45를 출력

eval 2 a b - + ; 67

eval 3 c + ; Error: c is not defined # 오류 메시지 출력

quit ; # 종료

* **알고리즘 및 자료구조**

Class Stack

* 생성자 초기화 items, value, dicitems(set함수에서 변수의 이름과 값 생성)
* Push -> 스택 push 기능
* Pop -> 스택 pop기능 + error1을 검출
* Eval -> 후위 수식 계산 함수 + 변수 있을 경우 딕셔너리 변수에서 키를 검색해서 없을 경우 오류 검출. 딕셔너리 변수 있을 경우 해당하는 value값으로 변환
* Set -> 후위 수식을 계산 해서 사용자가 [0]에 해당하는 변수에 값을 저장하고 딕셔너리 변수 생성 + eval 함수와 마찬가지로 변수는 해당하는 value값으로 변환, 변수 없을 경우 error 반환
* Evaluation -> 후위 수식을 계산하는 함수
* Switch -> 사용자가 입력한 수식을 변수와 정수와 연산자 따로 분리해 리스트를 만들어 주는 함수

Main function

* eval명령어 -> 후위수식을 계산하고 결과값 출력
* set명령어 -> 후위 수식을 계산하고 수식의 첫번째 변수에 딕셔너리 형태로 값 저장
* quit명령어 -> 프로그램 종료
* **느낌점**

스택 모델과 후위 수식 계산을 기계언어 처리하는 과정으로 기계어 처리하는 것처럼 프로그램을 만들어 볼 수 있는 프로그래밍에 대해 한 단계 성숙해지는 과정이었습니다. 두 번째 과제에 이어 명령어를 입력하고 수식을 계산하는 계산기 프로그램을 코딩해보면서 계산기 알고리즘과 스택 모델에 대해 시각적으로 이해 할 수 있는 시간이었습니다.

* **프로그램 코드**

import string  
# string.ascii\_lowercase # 소문자 abcdefghijklmnopqrstuvwxyz  
# string.ascii\_uppercase # 대문자 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
# string.ascii\_letters # 대소문자 모두 abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ  
# string.digits # 숫자 0123456789  
  
class Stack:  
 def \_\_init\_\_(self): # 스택 생성자 초기화 및 set함수에 사용할 딕셔너리 변수 생성  
 self.items = []  
 self.value = 0  
 self.dicitems = dict() # 클래스에서 딕셔너리 생성자 만들때 self.변수이름 = dict()  
  
 def push(self, e): # push 함수를 쓸경우 스택에 해당인자 push  
 self.items.append(e)  
  
 def pop(self): # pop에서 스택이 비어 있을 경우 indexError예외 처리하고 error1 반환  
 try:  
 return self.items.pop()  
 except:  
 print('error1')  
  
 def eval(self, command): # eval 명령어 양식: “eval 수식 ;”  
 oi = string.ascii\_letters # 영어 대소문자 다 저장함  
 x = []  
 for item in command: # command에는 eval 빼고 후위 수식만 들어 있음  
 if item in oi: # 영어 대소문자를 만날 경우는 변수로 인식함  
 if item in self.dicitems: # 그 변수가 딕셔너리 키에 있을 경우  
 x.append(str(self.dicitems[item])) # 키에 해당하는 value값 x리스트 에 append  
 else:  
 print("Error:", item, "is not defined") # 키(변수)를 못찾을 경우 error 반환  
 else:  
 x.append(item) # 변수(영어)가 아닐경우 정수나 상수이므로 x리스트에 append  
 eval\_result = self.evaluation(x) # 후위 수식을 계산  
 return eval\_result # 여기서, 수식은 후위 표기 수식임.  
 # eval 명령어 수행 결과: 수식을 계산하여 그 결과값을 출력.  
  
 def set(self, command):# set 명령어 양식: “set 변수 수식 ;”  
 command1 = command[1:] # command[0] 부분은 딕셔너리에 저장할 변수이므로 인덱스 1부터  
 oi = string.ascii\_letters # 후위 수식 계산  
 x = []  
 for item in command1: # command1에는 set 과 딕셔너리에 저장할 변수 빼고 후위 수식만 있음  
 if item in oi: # 후위 수식에서 변수를 찾음  
 if item in self.dicitems: # 변수가 딕셔너리 키 값에 있을 경우  
 x.append(str(self.dicitems[item])) # 키에 해당하는 value값 append  
 else:  
 print("Error:", item, "is not defined") # 변수를 못찾을 경우 error 반환  
 else:  
 x.append(item) # 변수(영어)가 아닐경우 정수나 상수이므로 x리스트에 append  
 self.dicitems[command[0]] = self.evaluation(x) # 저장할 변수에 후위 수식 결과값 반환  
 return self.dicitems  
 # 여기서, 변수는 알파벳으로 시작하는 문자열이고, 수식은 후위 표기 수식임.  
  
 def evaluation(self, expr): # expr : list  
 s = Stack()  
 op = '+-\*//%' # op = ('+','-','\*','/')  
 for item in expr: # 리스트의 원소를 하나씩 검사  
 if item in op: # 원소가 연사자일 경우 pop 두번해서 연산하고 다시 push 하는 과정  
 right\_opr = s.pop() # 첫번째 원소 pop 두 번째 원소 pop  
 left\_opr = s.pop() # operand: 피연산자, operator: 연산자  
 if item == '+':  
 s.push(left\_opr + right\_opr)  
 elif item == '-':  
 s.push(left\_opr - right\_opr)  
 elif item == '\*':  
 s.push(left\_opr \* right\_opr)  
 elif item == '//':  
 s.push(left\_opr // right\_opr)  
 elif item == '%':  
 s.push(left\_opr % right\_opr)  
 elif item == ';': # 원소가 세미콜론일 경우 for구문 break  
 break  
 else:  
 s.push(int(item)) # 원소가 연산자가 아닐 경우는 이문제에서는 정수 와 상수뿐이므로 push  
 if len(s.items) > 1: # 스택에 원소가 남아 있을 경우  
 return 'error2' # 오류임  
 else:  
 return s.pop() # 오류가 아닐경우 후위수식 결과값 pop해서 반환  
  
 def switch(self, exprStr): # 사용자의 입력을 공백 spilt해서 받아온 리스트를 정수 따로 연산자따로 변환해서 리스트화 하는 함수  
 x = []  
 op = '+', '\*', '%', '-', ';' # - 연산자는 - 정수부분 처리를 위해 따로 처리  
 oi = '0123456789' # // 연산자도 문자길이가 2이므로 따로 처리  
 val = '' # //와 - 연산자를 제외한 자머지 연사자를 처리하기 위한 변수  
 val1 = '' # // 연사자를 처리하기 위한 변수  
 for c in exprStr: # 리스트의 원소 한개씩 검사  
 if len(c) == 1: # 원소가 길이 한개 인 것들은 append함  
 x.append(c)  
 elif c[0] in '-': # -(피연산자) 0보다 작은 정수의 경우는 스트링 [0]검사해서 '-'있으면 append  
 x.append(c)  
 elif len(c) > 1: # 피연산자와 연산자가 붙어 있는 경우 떼어내는 부분  
 for s in c: # 한 글자씩 검사  
 if s in op: # 한글자가 연산자일 경우  
 x.append(val) # val변수는 연산자 이전에 숫자가 있을 경우 val에 저장하여  
 val = '' # 연산자를 만날경우 숫자 반환하고 다시 초기화  
 x.append(s) # 새로운 리스트 x에 append 하는 부분  
 elif s in oi: # oi는 숫자임  
 val += s # 숫자일 경우 val 변수에 string으로 이어 붙이기  
 elif s == '/': # // 연산자도 길이가 2이므로 문자 한개씩 검사하 하다가 //만날경우  
 val1 += '/' # 숫자 string 이어 붙이기와 같은 메커니즘으로 //가 될경우 append  
 if val1 == '//':  
 x.append(val1) # 새로운 리스트 x에 append합니다.  
 x.append(val) # 그냥 길이가 1이상인 숫자만 있을 경우 val에 저장된 숫자 append  
 val = ''  
 while '' in x:  
 x.remove('') # x리스트 안에 ['']원소는 모두 삭제  
 return x  
  
  
def main(): # main() 바깥에서 실행해도 됨. 그냥 파이썬 처럼 해석됨.  
 cal = Stack() # stack 클래스 불러옴  
  
 while True:  
 com = input().split() # com이라는 변수에 변수제거해서 spilt  
 command = cal.switch(com[1:]) # 명령어 빼고 나머지 후위수식만 변수에 저장  
 if com[0] == 'eval': # eval 명령어  
 print(cal.eval(command[:])) # 후위 수식 결과값 출력  
 elif com[0] == 'set':  
 cal.set(command[:])  
 elif com[0] == 'quit':  
 break  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()