프로그래밍 과제 4

이상윤 201802529

컴퓨터전자시스템공학전공

제출 일자 : 2022.04.13

1.

(1) 문제 기술

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: Linked 스택으로 구현한 스택을 사용한다.

입력: 후위 표기 수식이 한 줄에 주어진다.

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

(2) 코드

```
# Node 클래스 구현
class Node:
    def __init__(self, val):
        self.data = val
        self.link = None

# Linked 스택 클래스 구현
class LinkedStack:
    def __init__(self):
        self.top = None

    # 비어있는가?
    def isEmpty(self):
        return self.top == None

    # 삽입
```

```
    def push(self, val):
        newNode = Node(val)
        newNode.link = self.top
        self.top = newNode

    # 삭제 후 반환 (pop)
    def pop(self):
        if self.isEmpty():
            return None
        e = self.top.data
        self.top = self.top.link
        return e



# 덧셈 함수
def add(a, b):
    return a + b

# 뺄셈 함수
def sub(a, b):
    return a - b

# 곱셈 함수
def mul(a, b):
    return a * b

# 몫 반환 함수
def div_quo(a, b):
    return a // b
```

```
# 나머지 반환 함수
def div_rem(a, b):
    return a % b


# 후위 표기 수식을 받아 결과값을 계산하는 함수
def func(postfix_lst):
    # 후위 표기 수식의 각 요소를 순서대로 탐색
    for elem in postfix_lst:
        # 요소가 연산자라면, 스택에서 두 요소를 pop해 해당 연산 수행
        if elem == '+' or elem == '-' or elem == '*' or elem == '//' or elem ==
'%':
            op = elem
            opr2 = ls.pop()
            opr1 = ls.pop()

            # pop할 요소(수)가 없다면, 후위 표기 수식에 오류 존재
            if opr2 == None or opr1 == None:
                return None

            # math_func 딕셔너리를 활용하여 계산
            ls.push(math_func[op](opr1, opr2))

        # 맨 끝에 있는 ';'는 무시
        elif elem == ';':
            pass

        # 요소가 숫자라면, 스택에 push
        else:
            ls.push(int(elem))
```

```
    # 탐색이 끝났으면 스택엔 최종 결과값 하나만 있어야 한다.
    # 이를 pop해 반환
    ans = ls.pop()

    if ls.isEmpty:
        return ans
    else:
        # 요소에 ans 외 다른 요소가 있다면, 후외 표기 수식에 오류 존재
        return None


# main 함수 시작

# 계산을 도울 math_func 딕셔너리 정의
math_func = {
    '+' : add,
    '-' : sub,
    '*' : mul,
    '//' : div_quo,
    '%' : div_rem
}

# ls Linked 스택을 정의하고, 후위 표기 수식을 입력 받음
ls = LinkedStack()
postfix_lst = list(input().split())

# 입력 받은 후위 표기 수식을 func 함수로 넘겨 답 계산
ans = func(postfix_lst)

# 답이 None이라면 후위 표기 수식에 오류가 있다는 의미이므로 오류 출력, 그렇지 않으면
```

```
정답 출력
if ans == None:
    print("error")
else:
    print(ans)
```

(3) 자료구조 및 알고리즘 설명

후위 표기 수식을 계산하기 위해 스택을 활용하였습니다.

우선 후위 표기 수식 연산을 공백 기준으로 쪼개어 리스트에 저장하고, 순서대로 하나씩 탐색합니다.

만일 탐색 중인 값이 수(피연산자)라면, 이를 스택에 저장하고, 연산자라면, 스택에서 값 두 개를 pop한 후 '(늦게 pop한 값) (연산자) (먼저 pop한 값)'을 계산하여 스택에 다시 저장합니다.

위의 과정을 진행하는 중 스택이 비어 요소를 pop할 수 없거나, 과정이 끝나고 스택에 결과값 외 다른 요소가 남아있다면, 이는 후위 표기 수식에 오류가 있다는 의미이므로 None을 리턴하여 따로 처리해줍니다.

(4) 느낀점

문제를 풀면서 코드의 가독성을 높이기 위해 여러모로 고민했습니다. 그래서 정답을 계산하는 과정을 따로 함수로 만들었고, 계산 과정을 수행할 때 딕셔너리를 활용해 사칙연산을 수행하는 함수를 호출하도록 했습니다. 비록 테스트 케이스들 중 하나를 결국 통과시키지 못했지만, 그래도 조금이라도 더 나은 코드를 작성했다는 점을 위안으로 삼고 싶습니다.

3월에 조금 무리를 한 탓인지 시험 기간을 앞두고 슬럼프가 갑작스레 찾아온 바람에 2번 문제는 결국 코드를 완성하지 못하였습니다. 그래서 마음이 많이 착잡하지만, 다시 한 번 컨디션 관리 및 페이스 조절의 중요성을 실감하며 다가오는 중간고사를 잘 준비하기로 마음먹습니다.