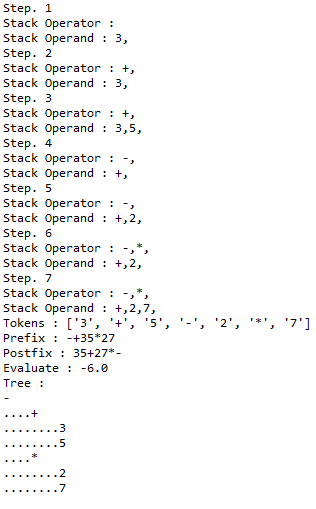
Homework 3

Calculator with Binary Tree

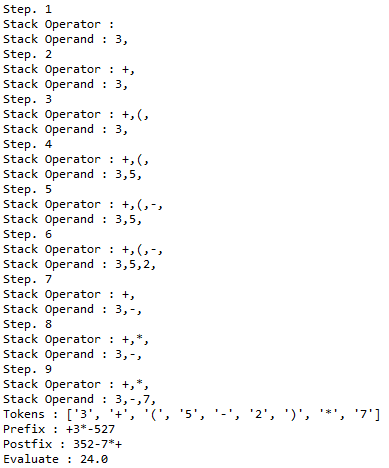
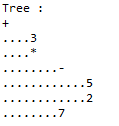
20150669 전예일

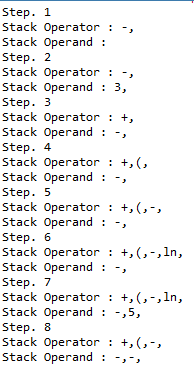
* Screenshots

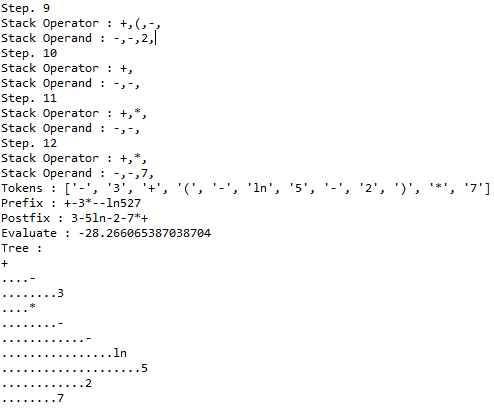
1. Testcase1 : line = '3 + 5 - 2 \* 7'



1. Testcase2 : line = '3 + ( 5 - 2 ) \* 7'

1. Testcase3 : line = '- 3 + ( - ln 5 - 2 ) \* 7'



* Reports

(Skeleton code에서 주어진 isDigit 함수는 사용하지 않았다)

Code는 homework3 introduction에 주어진 내용에 따라 binary Tree를 만들어가며 진행되게 되어있다. Code의 핵심내용이 되는 buildTree 함수를 구현하는 것이 중요한데 이것은 다음과 같이 구성되어 있다.

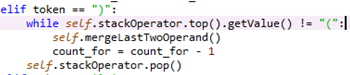


주어진 tokens을 받아서 digit인지 아닌지는 if token in priorities.keys(), if token == “(“, if token == “)” 등으로 operator 인지를 확인하고 그것이 아니면 digit로 간주하였다(isdigit함수를 사용하지 않음). Digit는 stackOperand에 추가하고 아닌 것은 introduction에 나온 대로



를 통해 stackOperator이 비었거나 stackOperator.top()의 priority가 현재 node의 priority보다 높낮으면 stackOperator에 추가되었으며 아니라면 mergeLastTwoOperand 함수를 통해 binaryTree의 형태로 합치게 된다. 이러한 진행이 모두 끝난 후에 stackOperator와 stackOperand에 남아있는 node들은 를 통해 stackOperand에 하나의 binaryTree로 만들어 지게 되고 그 Tree를 return하게 된다.

여기까지는 기본적인 내용이고 코딩을 통해 추가된 과정은 괄호 ‘(‘, ‘)’를 다루는 것과 unaryprioritiy에 관한 ‘-‘, ‘ln’을 다루는 내용이다. ‘(‘의 경우 stackOperator에 바로 추가해도 이상이 없으므로 추가되었고 이후에 ‘)’이 node로 들어오게 되면

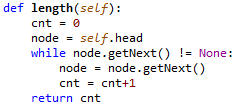


를 통해 stackOperator.top()이 ‘(‘이 될때까지 mergeLastTwoOperand 함수를 실행시키며 이후 ‘(‘ node를 stackOperator 에서 제거한다. 즉, 두 괄호 ‘(‘, ‘)’ 사이의 node들에 대하여 mergeLastTwoOperand를 실행한다.

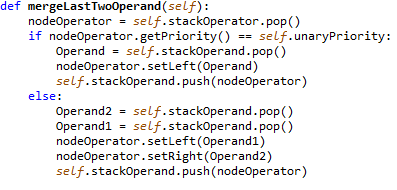
자연로그 ln의 경우  에서 unaryPriority로 설정이 되어 push된다(unaryPrioirity를 처리하는 내용은 mergeLastTwoOperand 함수에 포함되어 있다.

‘-‘ token의 경우 조금 복잡하다. 음수를 나타낼 수도 있고 빼기 기호를 나타낼 수도 있어서 이를 구분하는 방법이 필요한데 그를 위해 count\_for 이라는 변수를 사용한다. 정상적인 계산식에서는 digit, operator, digit, operator … (괄호 operator제외) 등 항상 digit가 operator보다 먼저 등장할 수 밖에 없다(operator는 2개의 digit을 필요로 하기 때문). 따라서 stackOperator 안의 operator의 숫자는 stackOperand보다 작거나 같아야만 한다. 괄호는 계산을 하지 않고 unarypriority의 ln, - 역시 그러하므로 이것을 제외한 +, -, \*, /, ^의 숫자를 count\_for을 통해 세어준다. 그리고

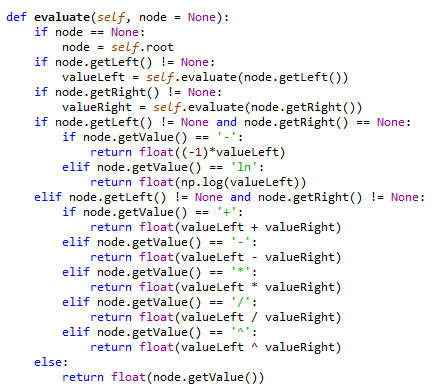
 에서 처럼 현재 token이’-‘ 일떄count\_for이 stackOperand의 element보다 같거나 많을 경우 이 ‘-‘는 음수를 나타내는 ‘-‘가 되므로 priority를 unarypriority로 설정해 주면 된다. (Stack class에 length 함수를 추가하여 element의 수를 계산할 수 있게 하였다.



함수 mergeLastTwoOperand 의 경우



이렇게 되어 2개의 Operand를 Operator 의 Left, Right child 로 만들어 주거나 ln과 음수를 나타내는 –의 경우에는 Left의 child로만 넣어주게 된다. 그리고 이것은



함수 evaluate에서 Left child만 있는 경우에 대하여 계산할 수 있게 하여 준다.

traversPreFix와 traversPostFix 역시 PreOrder, PostOrder처럼 방문 순서만 조절해 주면 원하는 결과값을 얻을 수 있다.

