CS 6301: Implementation of data structures and algorithms
Short Project 3: Recursive Descent Parser

Ver 1.0: Initial description (Fri, Feb 10).
Due: 11:59 PM, Fri, Feb 17.

- Submission procedure is same as the same as that of prior projects.
- For each group, only the last submission is considered. Earlier submissions are discarded.
- Your code must be of good quality, and pass all test cases to earn full credit.

Project Description

In this project, you will implement recursive descent parser discussed in the class.

Add the following method to Num class.

static Num evaluateExp(String expr): parse/evaluate the given expression and return the resulting number. You need to implement the recursive descent parser discussed in the class. Assume that the grammar is the same as the one we discussed in the class. If the given string is not valid, i.e., not a string that can be generated by the grammar, the method should throw an exception. The operands and operators in the input string may or may not be separated by empty space. For example, both "(3+4) * 5" and "(3 + 4) *5" are valid inputs.

The numbers in the expression can be of arbitrary length. Use Num. If there are '-' and/or '/' operators in the expression, throw unsupported operator exception.

<u>Use StringBuilder to tokenize the input string, as strings are immutable in Java.</u>

The following methods are optional.

- static String toPostfix(String expr): Implement shunting yard algorithm
- static Num evaluatePostfix(String[] expr): Evaluate an expression in postfix and return the resulting number.