CS 6301: Implementation of data structures and algorithms
Long Project 1: Integer arithmetic with arbitrarily large numbers

Ver 1.0: Initial description (Mon, Jan 27).
Due: 11:59 PM, Sun, Feb 23.

Max excellence credits: 1.0.

- 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.
- To earn an excellence credit implement correctly the method evaluateExp()**.**

Project Description

In this project, develop a program that implements arithmetic with large integers, of arbitrary size.

Code base: Java library: Lists, stacks, queues, sets, maps, hashing, trees. Do not use BigInteger, BigNum, or other libraries that implement arbitrary precision arithmetic. The starter code Num.java (in shared class folder).

Your task is to implement the class Num that stores and performs arithmetic operations on arbitrarily large integers. You must use the following data structure for representing Num: Array of long integers, where the digits are in the chosen base. In particular, do not use strings to represent the numbers. Each entry of the list stores exactly one long integer. The base is defined to be 10 in the starter code, but you may modify it. In the discussions below, we will use base = 10. For base = 10, the number 4028 is represented by the list: {8,2,0,4}.

It is mandatory to implement all methods other than method evaluateExp in the starter code. To earn excellence credit, implement evaluateExp(). Some of the methods in the starter code are:

- Num(String s): Constructor for Num class; takes a string s as parameter, with a number in decimal, and creates the Num object representing that number in the chosen base. **Note that, the string s is in base 10, even if the chosen base is not 10.** The string s can have arbitrary length.
- Num(long x): Constructor for Num class.
- String toString(): convert the Num class object into its equivalent string (in decimal). There should be no leading zeroes in the string.
- Num add(Num a, Num b): sum of two numbers a+b stored as Num.
- Num subtract(Num a, Num b): a-b
- Num product(Num a, Num b): product of two numbers a*b.

- Num power(Num x, long n): given an Num x, and n, returns the Num corresponding to x^n (x to the power n). Assume that n is a nonnegative number. Use divide-and-conquer to implement power using O(log n) calls to product and add.
- printList(): Print the base + ":" + elements of the list, separated by spaces.
- Num divide(Num a, Num b): Integer division a/b. Use divide-and-conquer or division algorithm. Return null if b=0.
- Num mod(Num a, Num b): remainder you get when a is divided by b (a%b). Assume that a is non-negative, and b > 0. Return null if b=0.
- Num squareRoot(Num a): return the square root of a (truncated). Use binary search. Assume that a is non-negative. Return null if b < 0.
- Num evaluatePostfix(String[] expr): evaluate the expression in postfix and return the resulting number.
- 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. Use StringBuilder to tokenize the input string, as strings are immutable in Java. Implementing this method correctly will earn you an excellence credit.