

**THE UNIVERSITY OF TEXAS AT DALLAS**

****

**LONG PROJECT 1**

**JAYAKARTHIGAYAN SRIDHARAN – jxs143730**

**SATHYANARAYANAN SRINIVASAN – sxs142031**

**SAAGARIKHA SRINIVASAN – sxs153930**

**SAHITHIREDDY ANDEM – sxa145430**

**CS 6301.002 Implementation of advanced data structures & algorithms**

**G19**

**Level 1**

In level 1, all numbers are non-negative integers. Implement the following methods:

1. XYZ(String s): Constructor for XYZ class; takes a string s as parameter, that stores a number in decimal, and creates the XYZ object representing that number. The string can have arbitrary length. In Level 1, the string contains only the numerals 0-9, with no leading zeroes.
2. XYZ(Long num): Constructor for XYZ class.
3. String toString(): convert the XYZ class object into its equivalent string (in decimal). There should be no leading zeroes in the string.
4. XYZ add(XYZ a, XYZ b): sum of two numbers stored as XYZ.
5. XYZ subtract(XYZ a, XYZ b): given two XYZ a and b as parameters, representing the numbers n1 and n2 repectively, returns the XYZ corresponding to n1-n2. If you have implemented negative numbers, return the actual answer. If not, then if the result is negative, it returns the XYZ for number 0.
6. XYZ product(XYZ a, XYZ b): product of two numbers.
7. XYZ power(XYZ a, long n): given an XYZ a, representing the number x and n, returns the BigNum 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.
8. printList(): Print the base + ":" + elements of the list, separated by spaces.

**Status: Implemented**

**Level 2 (EC: 0.7)**

Implement Level 1 and the following additional capabilities. Implement negative numbers, so that subtract returns the correct answer instead of 0 when the result is negative.

1. XYZ power(XYZ a, XYZ n): return a^n, where a and n are both XYZ. Here a may be negative, but assume that n is non-negative.
2. XYZ divide(XYZ a, XYZ b): Divide a by b result. Fractional part is discarded (take just the quotient). Both a and b may be positive or negative. If b is 0, raise an exception.
3. XYZ mod(XYZ a, XYZ b): remainder you get when a is divided by b (a%b). Assume that a is non-negative, and b > 0.
4. XYZ squareRoot(XYZ a): return the square root of a (truncated). Use binary search. Assume that a is non-negative.

**Status: Implemented**

#### Level 3 (EC: 0.3)

1. Shunting Yard Algorithm for
   1. Parenthesized expressions (...)
   2. Unary operators: factorial (!), and square root (~).
   3. Exponentiation (^), right associative.
   4. Product (\*), division (/), and, mod (%). These operators are left associative.
   5. Sum (+), and difference (-). These operators are left associative.

**Status: Implemented**

**Compiling and Running Instructions**

**Compiling Instruction:** javac LargeNumberArithmetic.java

**Running Instruction:** java LargeNumberArithmetic <Input File Name>.txt [Optional: Base Number 2 – 10,000]