**LP1: Integer arithmetic with arbitrarily large numbers**

**G97**

**Aim**:

The aim of the program is to implement arithmetic with large integers, of arbitrary size

**Operations Implemented:**

The following operations were implemented

Level 1:

1. BigNumber (String s): Constructor for BigNumber class; takes a string s as parameter, that stores a number in decimal, and creates the BigNumber object representing that number.
2. BigNumber (Long num): Constructor for BigNumber class.
3. String toString(): convert the BigNumber class object into its equivalent string (in decimal). Leading zeroes have been removed.
4. BigNumber add(BigNumber a, BigNumber b): sum of two numbers stored as BigNumber.
5. BigNumber subtract(BigNumber a, BigNumber b): given two BigNumber a and b as parameters, representing the numbers n1 and n2 repectively, returns the BigNumber corresponding to n1-n2. Subtract returns the correct answer even when the result is negative
6. BigNumber product(BigNumber a, BigNumber b): product of two numbers.
7. BigNumber power(BigNumber a, long n): given a BigNumber a, representing the number x and n, returns the BigNum corresponding to x^n (x to the power n). We have assume that n is a nonnegative number. Divide and conquer algorithm has been used.
8. printList(): Print the base + ":" + elements of the list, separated by spaces.

Level 2:

1. BigNumber power(BigNumber a, BigNumber n): return a^n, where a and n are both BigNumber. Here a may be negative, but n is assumed non-negative. Divide and conquer algorithm has been used.
2. BigNumber divide(BigNumber a, BigNumber b): Divide a by b result. Fractional part is discarded .Both a and b may be positive or negative. If b is 0, we have raise an exception.We have binary search algorithm to implement divide.
3. BigNumber mod(BigNumber a, BigNumber b): remainder you get when a is divided by b (a%b). The divide routine is invoked and the product of quotient and divisor is got and its is subtracted from the number.
4. BigNumber squareRoot(BigNumber a): return the square root of a (truncated. We have used binary search algorithm.

#### Level 2: Input/Output specification:

Instructions are divided into following Classes:

* Assignment Instructions
* Unary Arithmetic instructions
* Binary Arithmetic Instructions
* Display Instruction
* Branching instructions

**Instructions** is the base **abstract** class for the above instruction types

Each Instruction type must implement execute method:

Which performs the following activities

1. Perform its basic operation
2. Update the program counter to point to the next instruction (In case of branching instruction update program counter is updated based on the condition evaluation )

The following are the input output specification for our project.

The program takes input from stdin (console). The input is a sequence of lines. Each line has the line number as its first element. Lines are usually numbered 1,2,..., but do not assume that the ith line is numbered i. The input uses names of variables, which are always single letters (a,b,...). Each line has one of the following formats (after the line number). There are no unnecessary spaces in the input. The input has at most 1000 lines. You may assume that the input has no errors. [Comments starting with # are not part of the specification):

# Operations of the form var=something calculate the expression in

# something and assign it to the variable on the left

lineno var=NumberInDecimal # sets x to be that number

lineno var=var+var # sum of two numbers

lineno var=var\*var # product of two numbers

lineno var=var-var # first number minus second number

lineno var=var/var # first number divided by second number

lineno var=var%var # remainder of first number divided by second number

lineno var=var^var # power

lineno var=var! # factorial of number

lineno var=var~ # square root of number

lineno var # print the value of the variable to stdout (console)

lineno var?notzero:zero # if var value is not 0, then go to Line number notzero

# :zero is optional, if present, go to line zero, if var value is equal to 0

lineno var) # call printList() for the XYZ of the variable

Sample input:

1 x=999

2 y=8

3 z=x+y

4 z

5 a=x^y

6 a

7 z)

Its output is shown below, assuming that B = 100. Only lines 4, 6, and 7 produce output.

1007

992027944069944027992001

100:7 10

#### Level 3:

**We have implemented the Shunting Yard algorithm. The shunting yard algorithm first takes the infix expression and converts it to the postfix expression. For the infix to postfix part, the output queue and the stack have been used for the operations. The postfix expression is evaluated and the result is retrieved. Stack data structure has been used for evaluating the result accordingly.**

**But the shunting yard algorithm code has not been integrated with the driver code.We have just implemented and tested it for various outputs.**

**Result:**

The project ran successfully for all given inputs. We noticed that for multiplication , the performance of traditional method of multiplication performed better than Karatsuba’s algorithm and hence we have used the traditional multiplication in our project.