COSC4315: Lists and Infinite Precision Arithmetic

1 Introduction

You will create a program that can evaluate arithmetic operators with integer numbers having any number of digits. These numbers are an alternative to fixed size integers or floating point numbers that always have a maximum number of accurate digits (dependent on the size of CPU registers).

The fundamental data structure is a list, which is available in every programming language.

2 Input and output

The input is a regular text file, where each line is terminated with an end-of-line character(s). Each line will contain an arithmetic operation between two numbers. The program should display the input expression and the results, separated with =.

Input example

```
0*0
0+1
123456*2593
2*2000000000000000
2*3
1+10
10000000000000000+1
123456678901234567890+8765432109876543210
9999999999999999999+1
```

Output example

```
0*0=0
0+1=1
123456*2593=320121408
2*2000000000000000=4000000000000000
2*3=6
1+10=11
10000000000000000+1=10000000000000001
123456789012345678+8765432109876543210=9999999999999999999
9999999999999999999+1=100000000000000000000
```

3 Program input and output specification, main call

The main program should be called **infinitearithmetic**. The output should be written to the console (e.g.printf or cout), but the TAs will redirect it to create some output file. Call syntax at the OS prompt (notice double quotes):

```
infinitearithmetic "input=<file name>;digitsPerNode=<number>".
```

Assumptions:

- The file is a small plain text file (say < 10000 bytes); no need to handle binary files.
- Only integer numbers as input (no decimals!). Output number without leading zeroes.
- Operators: $+*$
- do not break an arithmetic expression into multiple lines as it will mess testing.

4 Requirements

- Recursive function are required. It is unacceptable to have loops (while/for) to process the list. Loops are acceptable only to read the input file, but even then recursive functions are preferred.
- Lists are required to store the long integers. A program using arrays to store long numbers will receive a failing grade (below 50). However, arrays for parameters or other auxiliary variables are acceptable.
- Correctness is the most important requirement: TEST your program with many expressions. Your program should not crash or produce exceptions.
- Breaking a number into a list of nodes. Each node will store the number of digits specified in the parameters. Notice it is acceptable to "align" digits after reading the entire number so that that the rightmost node (end) has all the digits.
- The list needs to be passed as argument and/or return value, but cannot be a global variable. The program must be prepared to handle zeroes or simple errors like spaces or missing numbers extra credit for developing both forward and backward recursion starting from either side of the list.