COSC4315: Adding Infinite Integers

1 Introduction

You will create a program that can evaluate addition with integer numbers having any number of digits. Addition will be expressed as a function, not with the usual + operator. 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 programming language is: Python. The operating system: your source code will be tested on a Linux server with Python3, but you can develop it on any OS.

Notice you cannot use built-in integers with infinite length (i.e. Python int of unbounded length). Therefore, you are expected to program it with a list of 32 bit integers (i.e. int) with a maximum length of 40 digits. The fundamental data structure is a list, which is available in every programming language. You can use array lists or linked lists. The number of digits per node in the list should be between 1 and 8 (your choice, notice a 32 bit int allows 9 digits).

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 text file

Notice the first and last lines with = are shown to indicate where the file start and ends (they are not part of the input).

```
========================================
add(0,0)
add(1,2)
add(10000000000000000,1)
add(12345667890123456789,8765432109876543210)
add(999999999999999999,1)
========================================
```

Output example

Notice there are no spaces, other separators or more than one expression per line. Do not format the integers in a different way. Do not show the digits in each node separated by spaces or commas. Avoid including any other output strings as this will hinder automated testing. This is an example with the required format.

```
========================================
add(0,0)=0
add(1,2)=3
add(10000000000000000,1)=10000000000000001
add(12345667890123456789,8765432109876543210)=21111099999999999999
add(999999999999999999,1)=1000000000000000000
========================================
```

3 Program input and output specification, main call

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

```
python3 infint.py "input=<file name>"
```

Specification and assumptions:

- The input file is a required parameter. Notice the input string has ”input=” because in future programs there will be 2 or more input parameters. This file is a plain text file (say < 10000 bytes); no need to handle binary files.
- Addition is indicated with add().
- There is only one addition per line (very simple).
- Only integer numbers as input (no decimals!, no decimal point, no sign). Produce output number without leading zeroes.
- Main function: add(a,b) where a,b are integers

Example of program call:

```
python3 infint.py "input=add.txt"
```

Examples of incorrect program calls of python program:

```
python3 infint.py "add.txt"

python3 infint.py add
```

4 Requirements

- Lists are required to store the infinite integers.
- The addition algorithm is the one you learned in elementary school, from right to left. The main difference is that you will do it longer integers.
- You can develop an iterative or a recursive algorithm. In your comments specify your choice and the number of digits.

- 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 Notice it is acceptable to "align" digits after reading the entire number so that that the rightmost node (least significant digits) has all the digits.
- In the source code 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.
- Grading: this program will be graded PASS/FAIL. Any student getting FAIL or not submitting the program cannot stay in the course.