

Practice Exercise #38: Big Numbers

http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html

Objective:

Recursion on a linked list.

Task statement:

This is Take-home Lab #3 Exercise 2 in AY2012/2013 Semester 2. You are to repeat the exercise, but this time using recursion to sum the two big numbers recursively.

Chin is an undergraduate in the Department of Mathematics working on his final year project. His research on prime numbers requires the handling of large numbers. However, he discovered that Java primitives like integers do not support large numbers. In fact, the biggest integer that we can store in a variable of type `int` is $2^{31}-1$ on a 32-bit processor.

As a close friend of Chin, you vaguely recall that this problem can be solved with linked lists.

You will hence develop a way for Chin to input large numbers and output the sum of these numbers.

You are to create a Java class named **Digit** to represent a node. The attributes are the digit itself and reference to the next node in the list.

You will also need to write a Java class named **BigNumber** which contains a **sumRecursive()** method that takes in two big numbers (both are **BigNumber** objects) and return their sum (a `String` object).

Hint: Use each node to store a digit. Think about how an addition operation is performed on two numbers.

Important: You are **not allowed** to use the **BigInteger** (or equivalent) class, or the **LinkedList** class provided by Java. Doing so violates the objective of this exercise and hence your program will be deemed incorrect.

Input

The first line of the input contains a positive number **L** ($1 \leq L \leq 20$) which determines the number of addition operations to be performed. Each successive line contains two large positive integers **M** and **N** ($0 \leq M+N < \infty$) separated by a space.

(In your program, you should use more descriptive variable names instead of **L**, **M** and **N** and follow Java naming convention.)

Output

The output contains the sum of each **M+N**.

Sample Input #1

1

900139190258102985175475477 185120982701967121286128612

Sample Output #1

1085260172960070106461604089

Sample Input #2

3

012948190285125891275182612 12861892679161261826211
12961926190268126910261126 198491289512812961261261
858869126948395893001361261 129841209869182609128602186012657

Sample Output #2

12961052177805052537008823
13160417479780939871522387
129842068738309557524495187373918

Sample Input #3

2

6598034892389518295812098519285 1298591809435943758934759345436
0 0

Sample Output #3

7896626701825462054746857864721
0