Practice Exercise #35: Big Numbers

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

Objective:
Recursion on 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 an **add()** method that takes in two big numbers (both are String objects) and return their sum (also 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 **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
129619261902681269102611 26 198491289512812961261261
858869126948395893001361261 1298412098691826091286021860126 57
```

Sample Output #2

```
1296105217780505253700882 3
131604174797809398715223 87
129842068738309557524495187373918
```

Sample Input #3

```
2
659803489238951829581209851928 5 129859180943594375893475934543 6
0 0
```

Sample Output #3

```
789662670182546205474685786472 1
0
```