**第三次作业**

提交期限：2022年12月20日

在C, C++, Java等编程语言中整数的长度受到硬件平台限制，大部分为32位或者64位，即整数的范围为 — 或者—。运算中结果超出最大或最小整数的范围会导致整数溢出。譬如 C++中，2147483647+1 的输出为 -2147483648。

Python语言摆脱了操作系统和硬件的限制，内部实现了无限长度的整数运算。所以python可以直接计算2\*\*100 + 3\*\*50的值。

>>> 2\*\*100+3\*\*50

1267651318126217093349291975625

Python是如何实现无限长度的整数的呢？这是本作业的主要内容。一个可行思路是：用一个list来表示任意长度一个整数，然后模拟整数的竖式运算：

例如 x = 123456789，用listx = [1, 2, 3, 4, 5, 6, 7, 8, 9]

例如 y = 7777777，用listy = [7, 7, 7, 7, 7, 7, 7]

如果我们希望计算x+y的值，可以通过listx和listy来模拟整数加法，从而实现任意长度整数的加法运算：

listx=[1, 2, 3, 4, 5, 6, 7, 8, 9] 1 2 3 4 5 6 7 8 9

listy=[7, 7, 7, 7, 7, 7, 7] +) 7 7 7 7 7 7 7

对位相加 1 2 10 11 12 13 14 15 16

进位 1 3 1 2 3 4 5 6 6

最后的结果用list表示

[6, 6, 5, 4, 3, 2, 1, 3, 1]

转换为整数为 131234566

**注意：用列表表示一个整数，两个整数对齐时要注意，在上面的例子中整数最低位在list[0]。**

**本次作业的目的**

1. 实现用list模拟整数进行+，-，\*, //, \*\*. 其中\*\*的指数是用普通非负整数表示即可。
2. 完成代码，并提交一份单独的技术报告，说明每个函数的功能。

**包含的函数**

由于“我们的系统”不支持大整数，所以大整数是通过string来表示的

1234567890987654321 “1234567890987654321”

作业中**必须实现如下函数（不要改变如下函数头的定义）**：

def add(str1, str2): return str

def sub(str1, str2): return str

def mul(str1, str2): return str

def div(str1, str2): return str1，str2 (商，余数)

def pow(str1, n): return str #n为普通非负整数，不需要用list表示

即，实现用str表示的大整数的基本运算，结果也用str表示。

推荐的实现方法：将大整数从str转换为list，然后用list进行相关操作。（ppt上面有个参考代码，仅供参考）

def str2list(“123434444444”) [1,2,3,4,3,4,4,4,4,4,4,4] #怎么处理+，-符号？

def list2str([1,2,3,4,3,4,4,4,4,4,4,4]) “123434444444”

def add\_list(lst1, lst2): return list

def sub\_list(list1, list2): return list

def mul\_list(list1, list2): return list

def div(list1, list2): return list1，list2

def pow(list1, n): return list #n为普通非负整数，不需要用list表示

**测试**

请用你的函数**计算并输出**如下表达式的运算结果 (本测试占据一定比例分数)

22222222222222+8773849905050505

11111111-9877344555

345676778778-222222

123456\*789

8773849905050505//123

2\*\*66

2\*\*100+3\*\*50

2\*100 +123456\*789-8773849905050505//123

**（注：直接调用函数add, sub等计算出上面的表达式即可，不用写程序自动转换为你写的函数。）**

1. 本项目不强求用class来写大整数
2. 可以考虑实现Class中的+,-,\*,//等重载
3. 代码雷同：抄的零分，被抄的如果是主动提供也会有责任

目前为止，大家都写了一定量的代码了。在写代码的时候，注意一些基本原则：

1. 尽可能用函数，如果一个代码块包含几个功能，建议用函数包装。尽可能减少代码的逻辑复杂度，重用代码，降低出bug的可能性。
2. 多重循环不要超过2轮，超过就使用函数, 使得代码尽可能清晰。
3. 编码的时候，变量名、空格等等遵守python规范

>>> import this

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.