Шишмаков Владимир Олегович, гр. №5130201/30101

Программный модуль 1

1 Описание

Название: Реализация длинной арифметики для целых чисел.
Дано:

- M — целое число разрядов.
- N — целое число, характеризующее систему счисления.
- a, b — числа, над которыми производятся операции.

Требуется: Реализовать операции сложения, вычитания, умножения и целочисленного деления над целыми числами.
Ограничения:

- $N > 0$
- $M > 0$
- $a \geq 0$
- $b \geq 0$

Спецификация:

№	Вход	Выход	Реакция программы
1	$a = 15000000000; b = 1000000000; M = 10; N = 2$	a = [0, 0] — 0; b = [0, 0] — 0; Sum: [0, 0] — 0; Sub: [0, 0] — 0; Mul: [0, 0] — 0; Div: [0] — 0;	Программа принимает на вход числа a и b приводит их в формат длинной арифметики и выполняет над ними 4 операции. Выводит числа a и b, а также результаты операций над ними
2	$a = 2441; b = 2111; M = 10; N = 2$	a = [4, 1] — 41; b = [1, 1] — 11; Sum: [5, 2] — 52; Sub: [3, 0] — 30; Mul: [5, 1] — 51; Div: [0, 3] — 3;	Программа принимает на вход числа a и b приводит их в формат длинной арифметики и выполняет над ними 4 операции. Выводит числа a и b, а также результаты операций над ними
3	$a = 2441; b = 0; M = 10; N = 2$	a = [4, 1] — 41; b = [0, 0] — 0; Sum: [4, 1] — 41; Sub: [4, 1] — 41; Mul: [0, 0] — 0; Div: [0] — 0;	Программа принимает на вход числа a и b приводит их в формат длинной арифметики и выполняет над ними 4 операции. Выводит числа a и b, а также результаты операций над ними

2 Исходный код

```
M = 10
N = 2


def normalize(num, M=None, N=None):
    while len(num) > 1 and num[0] == 0:
        num = num[1:]
    return num


def to_int(num, M):
    num = normalize(num)
    result = 0
    for digit in num:
        result = result * M + digit
    return result


def from_int(num, M, N):

    mod = M ** N
    num %= mod

    digits = []
    for _ in range(N):
        digits.append(num % M)
        num //= M
    digits.reverse()
    return digits


def compare(first_value, second_value):
    a = normalize(first_value[:])
    b = normalize(second_value[:])
    if len(a) > len(b):
        return 1
    if len(a) < len(b):
        return -1
    for x, y in zip(a, b):
        if x > y:
            return 1
        if x < y:
            return -1
    return 0


def sum(first_value, second_value, M, N):
    a = first_value[-N:]
    b = second_value[-N:]
    max_len = max(len(a), len(b))
    a = [0] * (max_len - len(a)) + a
    b = [0] * (max_len - len(b)) + b

    carry = 0
    res = [0] * (max_len + 1)
    for i in range(max_len - 1, -1, -1):
        s = a[i] + b[i] + carry
        res[i + 1] = s % M
        carry = s // M
    res[0] = carry

    res = res[-N:]
```

```
    return res


def sub(first_value, second_value, M, N):
    a = first_value[-N:]
    b = second_value[-N:]
    max_len = max(len(a), len(b))
    a = [0] * (max_len - len(a)) + a
    b = [0] * (max_len - len(b)) + b

    res = [0] * max_len
    borrow = 0
    for i in range(max_len - 1, -1, -1):
        val = a[i] - b[i] - borrow
        if val < 0:
            val += M
            borrow = 1
        else:
            borrow = 0
        res[i] = val

    if borrow == 1:
        add_back = [M - 1] * max_len
        carry = 1
        for i in range(max_len - 1, -1, -1):
            val = res[i] + add_back[i] + carry
            res[i] = val % M
            carry = val // M

    res = res[-N:]
    return res


def mul_small(a, k, M, N):
    if k == 0 or a == [0]:
        return [0] * N

    a = a[-N:]
    res = [0] * (len(a) + 1)
    carry = 0
    for i in range(len(a) - 1, -1, -1):
        prod = a[i] * k + carry
        res[i + 1] = prod % M
        carry = prod // M
    res[0] = carry

    res = res[-N:]
    return res


def times(first_value, second_value, M, N):
    a = first_value[-N:]
    b = second_value[-N:]
    if a == [0] * len(a) or b == [0] * len(b):
        return [0] * N

    res = [0] * (2 * N)
    len_a = len(a)
    len_b = len(b)

    for i in range(len_a - 1, -1, -1):
        carry = 0
        for j in range(len_b - 1, -1, -1):
            idx = i + j + 1
            total = res[idx] + a[i] * b[j] + carry
```

```
            res[idx] = total % M
            carry = total // M
        res[i] += carry

    res = res[-N:]
    return res


def div(first_value, second_value, M, N):
    if compare(second_value, [0]) == 0:
        return [0] # division by zero

    a = first_value[-N:]
    b = second_value[-N:]
    a = normalize(a)
    b = normalize(b)
    if compare(a, b) < 0:
        return [0] * N

    quotient = []
    remainder = [0]
    for digit in a:

        remainder = remainder + [digit]
        remainder = normalize(remainder)

        lo, hi = 0, M - 1
        q = 0
        while lo <= hi:
            mid = (lo + hi) // 2
            prod = mul_small(b, mid, M, N)
            if compare(prod, remainder) <= 0:
                q = mid
                lo = mid + 1
            else:
                hi = mid - 1
        quotient.append(q)
        if q:
            remainder = sub(remainder, mul_small(b, q, M, N), M, N)

    quotient = quotient[-N:]
    return quotient


a = from_int(2441, M, N)
b = from_int(0, M, N)

print(f"a = {a} --- {to_int(a, M)}")
print(f"b = {b} --- {to_int(b, M)}")

print("Sum:", sum(a, b, M, N), "---", to_int(sum(a, b, M, N), M))
print("Sub:", sub(a, b, M, N), "---", to_int(sub(a, b, M, N), M))
print("Mul:", times(a, b, M, N), "---", to_int(times(a, b, M, N), M))
print("Div:", div(a, b, M, N), "---", to_int(div(a, b, M, N), M))
```