# СПЕЦІАЛЬНІ РОЗДІЛИ ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ

## Лабораторна 2

Терпило Софія

ФБ-06

## Завдання до комп’ютерного практикуму:

А) Доопрацювати бібліотеку для роботи з m-бітними цілими числами, створену на комп’ютерному практикумі No1, додавши до неї такі операції:

1)обчислення НСД та НСК двох чисел;

2) додавання чисел за модулем;

3) віднімання чисел за модулем;

4) множення чисел та піднесення чисел до квадрату за модулем;

5) піднесення числа до багаторозрядного степеня d по модулю n

## Хід роботи:

```python
# LAB 2

def odd_or_not(num1, hex_alphabet, base):
    div_result = long_div(num1, '2', hex_alphabet, base)
    if div_result[1] == '':
        return True
    else:
        return False


def long_gcd(num1, num2, hex_alphabet, base):
    result = '1'
    while odd_or_not(num1, hex_alphabet, base) and odd_or_not(num2, hex_alphabet, base):
        num1 = long_div(num1, '2', hex_alphabet, base)[0]
        num2 = long_div(num2, '2', hex_alphabet, base)[0]
        result = long_mul(result, '2', hex_alphabet, base)
    while odd_or_not(num1, hex_alphabet, base) and not odd_or_not(num2, hex_alphabet, base):
        num1 = long_div(num1, '2', hex_alphabet, base)[0]
    while num2 != '0':
        while odd_or_not(num2, hex_alphabet, base):
            num2 = long_div(num2, '2', hex_alphabet, base)[0]
        if long_compare(num1, num2, hex_alphabet) == 1: # num1 > num2
            num1, num2 = num2, num1
        num2 = long_sub(num2, num1, hex_alphabet, base)
    result = long_mul(result, num1, hex_alphabet, base)
    return result.lstrip('0')
```

```python
def long_lcm(num1, num2, hex_alphabet, base):
    mul_result = long_mul(num1, num2, hex_alphabet, base)
    gcd = long_gcd(num1, num2, hex_alphabet, base)
    div_result = long_div(mul_result, gcd, hex_alphabet, base)[0]
    return div_result


def barrett_reduction(num, mod, hex_alphabet, base):
    b_2k = long_pow('10', convert_to_hex(len(mod) * 2), hex_alphabet, base)
    mu = long_div(b_2k, mod, hex_alphabet, base)[0]
    k = len(mod)
    n_len = len(num)
    q = num[:n_len - k - 1]
    q = long_mul(q, mu, hex_alphabet, base)
    q = q[:len(q) - k + 1]
    r = long_mul(q, mod, hex_alphabet, base)
    r = long_sub(num, r, hex_alphabet, base)
    r = r.lstrip('0')
    while long_compare(r, mod, hex_alphabet) == 1:
        r = long_sub(r, mod, hex_alphabet, base)
    return r.lstrip('0')
```

```python
def long_op_mod(num1, num2, mod, operation, hex_alphabet, base):
    op_result = ''
    if operation == 'add':
        op_result = long_add(num1, num2, hex_alphabet, base)
        print('sum: ', op_result)
    elif operation == 'sub':
        op_result = long_sub(num1, num2, hex_alphabet, base)
        print('dif: ', op_result)
    elif operation == 'mul':
        op_result = long_mul(num1, num2, hex_alphabet, base)
        print('mul: ', op_result)
    if op_result != 'error':
        mod_result = barrett_reduction(op_result, mod, hex_alphabet, base)
    else:
        mod_result = 'error'
    return mod_result


def long_pow_barret(num, p, mod, hex_alphabet, base):
    p = convert_to_binary(p)
    product = '1'
    for digit in range(len(p)):
        if p[digit] == '1':
            product = long_mul(product, num, hex_alphabet, base)
        if digit != len(p) - 1:
            product = long_mul(product, product, hex_alphabet, base)
        print('product = ', product)
        product = barrett_reduction(product, mod, hex_alphabet, base)
    return product
```

## Контроль коректності:

```python
# TEST: (a + b) * c mod(n) = c * a + c * b mod(n)
sum = long_add(a, b, hex_alphabet, base)
prod = long_op_mod(sum, c, n, 'mul', hex_alphabet, base)
print('(a + b) * c mod n =', prod)

prod_2 = long_mul(a, c, hex_alphabet, base)
prod_3 = long_mul(c, b, hex_alphabet, base)
sum_2 = long_op_mod(prod_2, prod_3, n, 'add', hex_alphabet, base)
print('a * c + c * b mod n =', sum_2)

if prod == sum_2:
    print('TEST : (a + b) * c mod n = a * c + c * b mod n => passed')
```

```
C:\Users\User\PycharmProjects\srom_lab1_2\venv\Scripts\python.exe C:
(a + b) * c mod n = 3D1D4D1FB64ED6184CF50B1A68293E5751698F812852A2C0
a * c + c * b mod n = 3D1D4D1FB64ED6184CF50B1A68293E5751698F812852A2
TEST : (a + b) * c mod n = a * c + c * b mod n => passed

Process finished with exit code 0
```

## Час виконання операцій:

```
addition => 0.19131600000000004
multiplication => 1.0379357999999999
```

На жаль операція піднесення до степеня надзвичайно повільна(((