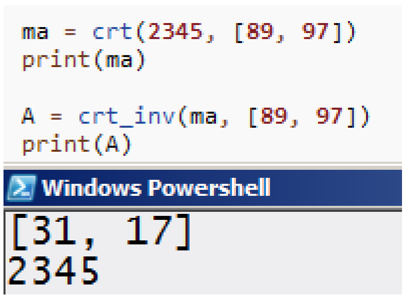
Задание 1, 2, 3.

```python
def crt(A, m1mk):
    '''
    По китайской теореме об остатках
    Возвращаем кортеж A = (a1, a2, ..., an)
    где ai = A mod mi, mi принадлежит m1mk
    '''
    return [A % m for m in m1mk]    # ma

def crt_inv(ma, m1mk):
    '''
    Возвращаем число A из Z_MZ, по представлению числа ma
    '''
    M = [m1mk[j] * m1mk[j+1] for j in range(len(m1mk) - 1)][0]
    Mi = [M / m for m in m1mk]
    Mi_inv = [get_inverse_eea(Mi[j], m1mk[j]) for j in range(len(m1mk))]
    c = [Mi[j] * Mi_inv[j] for j in range(len(Mi))]
    A = [(ma[j] * c[j] + ma[j+1] * c[j+1]) % M for j in range(len(c) - 1)][0]

    return int(A)
```

Результат:



```python
ma = crt(2345, [89, 97])
print(ma)

A = crt_inv(ma, [89, 97])
print(A)
```

Windows Powershell

```
[31, 17]
2345
```

Задание 4.

Переформулируем задачу: Нужно найти число, которое при делении по модулю на 5, 8 и 19 дает остатки 1, 2 и 3, соответственно.

```python
for x in range(1000):
    if x % 5 == 1 and x % 8 == 2 and x % 19 == 3:
        print(x)
```

Ответ: 706.

Задание 5.

Найдите решение системы

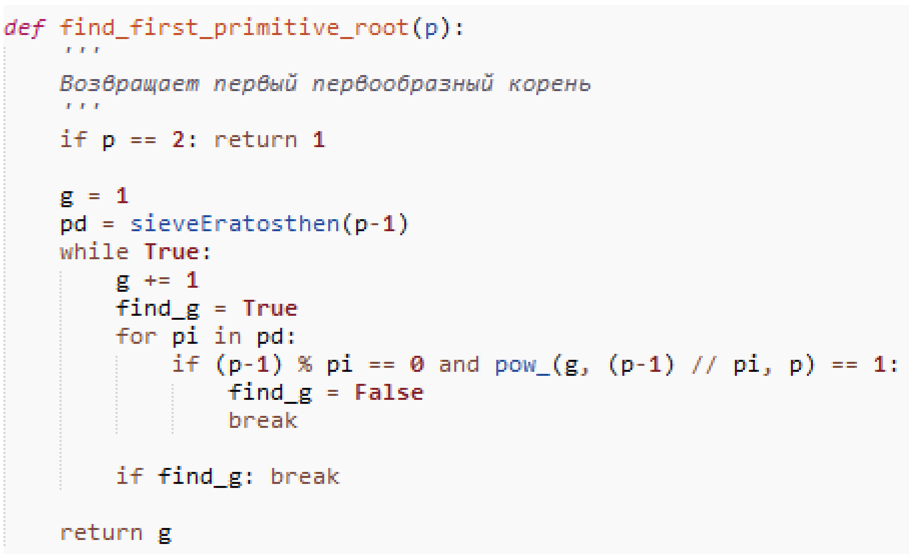
$$\begin{cases} x \equiv 3 \pmod{5}, \\ x \equiv 7 \pmod{9}. \end{cases}$$

Можно решать двумя способами:

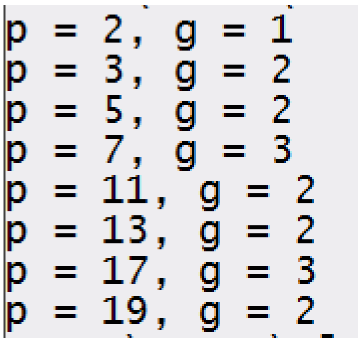
```python
for x in range(45):
    if x % 5 == 3 and x % 9 == 7:
        print(x)

x = crt_inv([3, 7], [5, 9])
print(x)
```

Ответ: x = 43.

Задание 6.

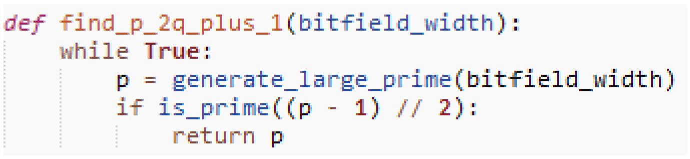


```python
def find_first_primitive_root(p):
    '''
    Возвращаем первый первообразный корень
    '''
    if p == 2: return 1

    g = 1
    pd = sieveEratosthen(p-1)
    while True:
        g += 1
        find_g = True
        for pi in pd:
            if (p-1) % pi == 0 and pow_(g, (p-1) // pi, p) == 1:
                find_g = False
                break

        if find_g: break

    return g
```

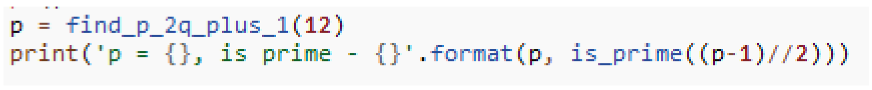


```
p = 2, g = 1
p = 3, g = 2
p = 5, g = 2
p = 7, g = 3
p = 11, g = 2
p = 13, g = 2
p = 17, g = 3
p = 19, g = 2
```

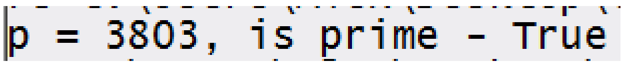
Задание 7.



```python
def find_p_2q_plus_1(bitfield_width):
    while True:
        p = generate_large_prime(bitfield_width)
        if is_prime((p - 1) // 2):
            return p
```

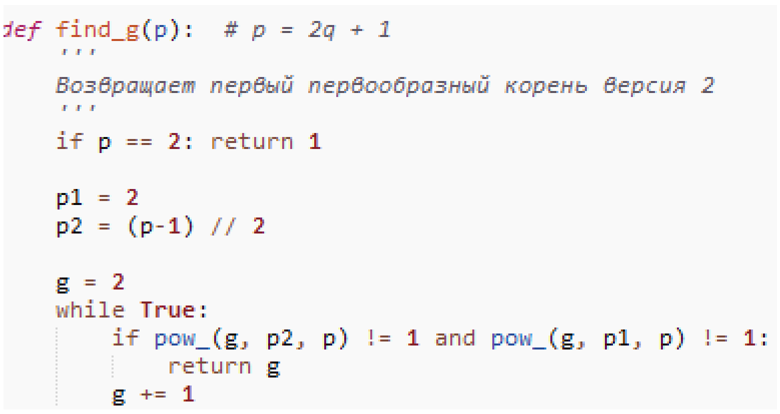


```python
p = find_p_2q_plus_1(12)
print('p = {}, is prime - {}'.format(p, is_prime((p-1)//2)))
```

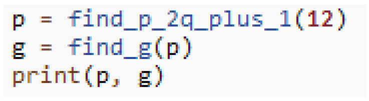


```
p = 3803, is prime - True
```

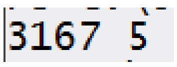
Задание 8.



```
def find_g(p):  # p = 2q + 1
    '''
    Возвращаем первый первообразный корень версия 2
    '''
    if p == 2: return 1

    p1 = 2
    p2 = (p-1) // 2

    g = 2
    while True:
        if pow_(g, p2, p) != 1 and pow_(g, p1, p) != 1:
            return g
        g += 1
```

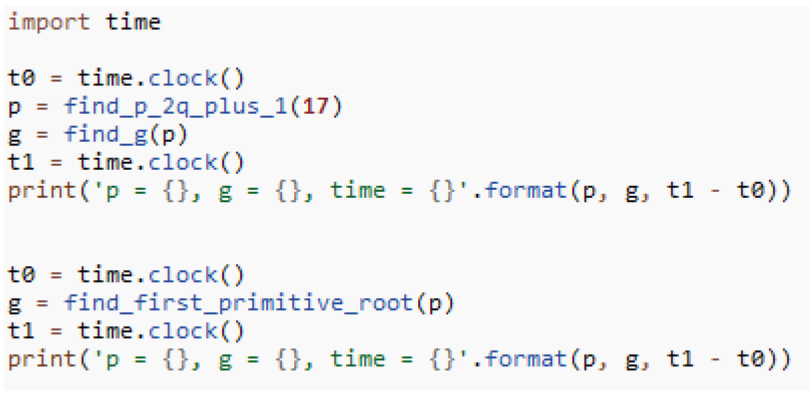


```
p = find_p_2q_plus_1(12)
g = find_g(p)
print(p, g)
```

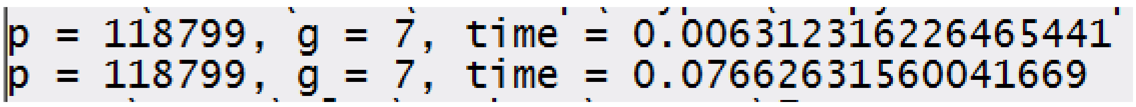


```
3167 5
```

Задание 9.

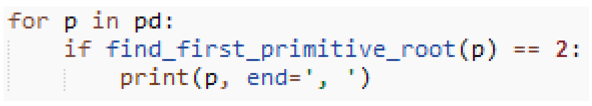


```
import time

t0 = time.clock()
p = find_p_2q_plus_1(17)
g = find_g(p)
t1 = time.clock()
print('p = {}, g = {}, time = {}'.format(p, g, t1 - t0))


t0 = time.clock()
g = find_first_primitive_root(p)
t1 = time.clock()
print('p = {}, g = {}, time = {}'.format(p, g, t1 - t0))
```

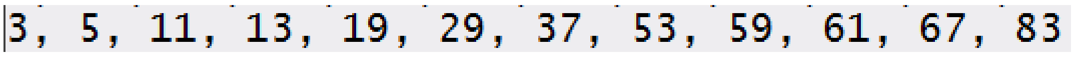


```
p = 118799, g = 7, time = 0.006312316226465441
p = 118799, g = 7, time = 0.07662631560041669
```

Задание 10.

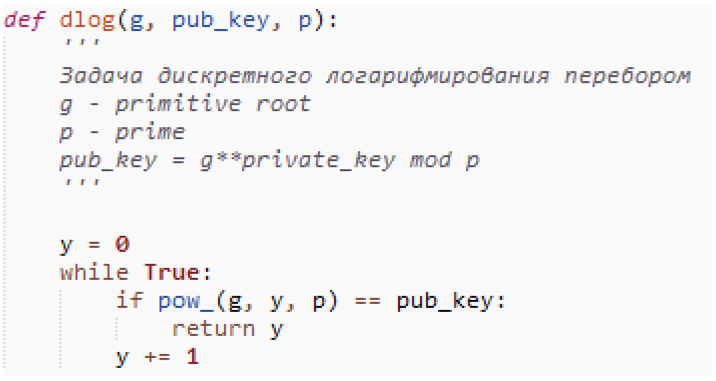


```
for p in pd:
    if find_first_primitive_root(p) == 2:
        print(p, end=', ')
```

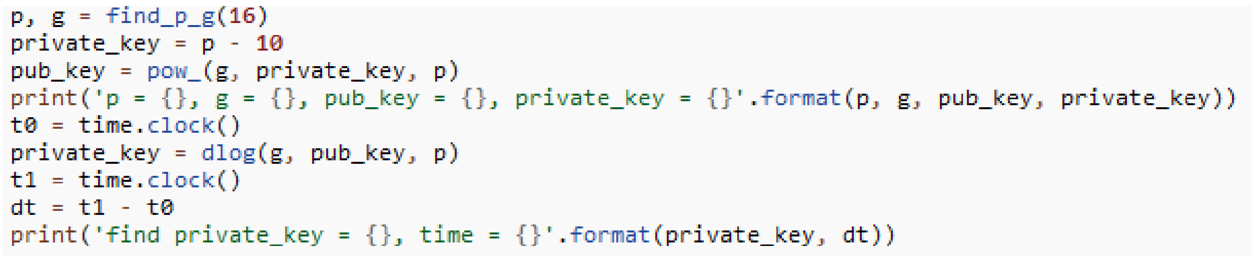


```
3, 5, 11, 13, 19, 29, 37, 53, 59, 61, 67, 83
```

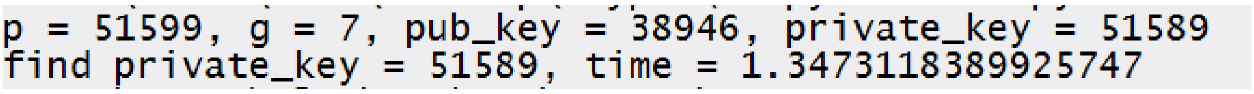
Задание 11.



```python
def dlog(g, pub_key, p):
    '''
    Задача дискретного логарифмирования перебором
    g - primitive root
    p - prime
    pub_key = g**private_key mod p
    '''

    y = 0
    while True:
        if pow_(g, y, p) == pub_key:
            return y
        y += 1
```



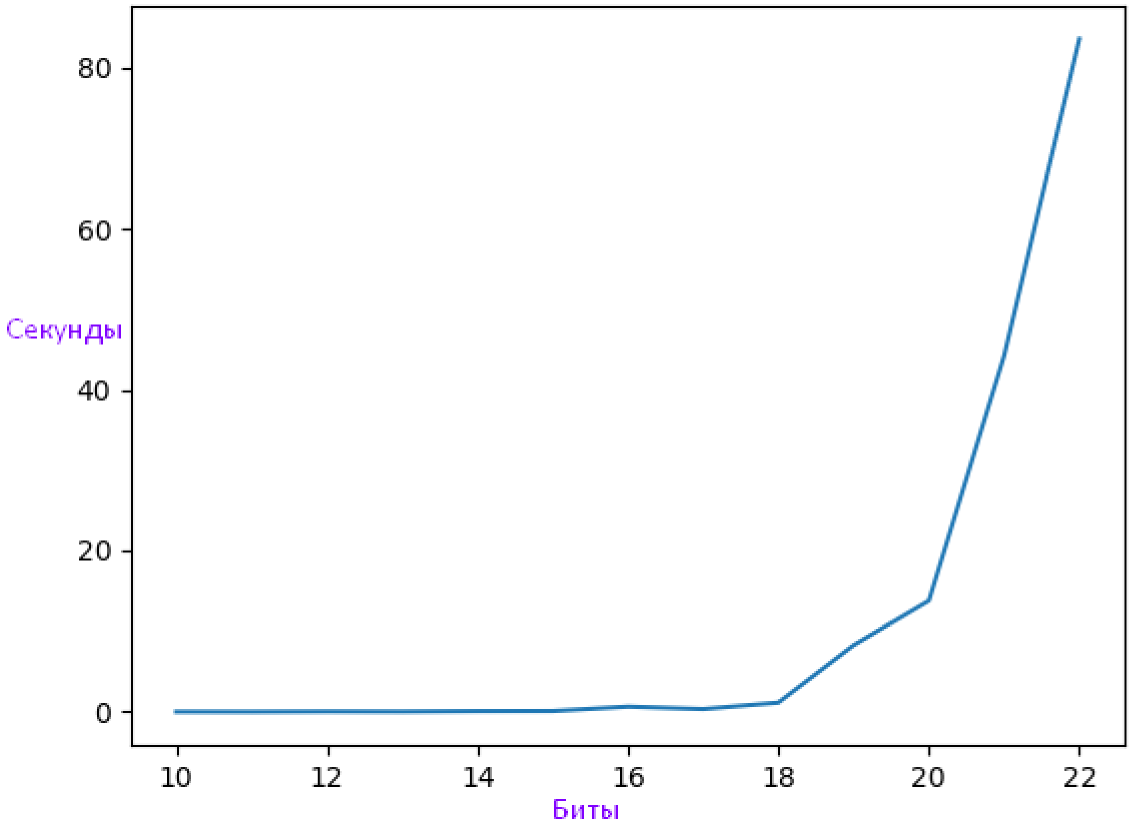
```python
p, g = find_p_g(16)
private_key = p - 10
pub_key = pow_(g, private_key, p)
print('p = {}, g = {}, pub_key = {}, private_key = {}'.format(p, g, pub_key, private_key))
t0 = time.clock()
private_key = dlog(g, pub_key, p)
t1 = time.clock()
dt = t1 - t0
print('find private_key = {}, time = {}'.format(private_key, dt))
```



```
p = 51599, g = 7, pub_key = 38946, private_key = 51589
find private_key = 51589, time = 1.3473118389925747
```

Задание 12.

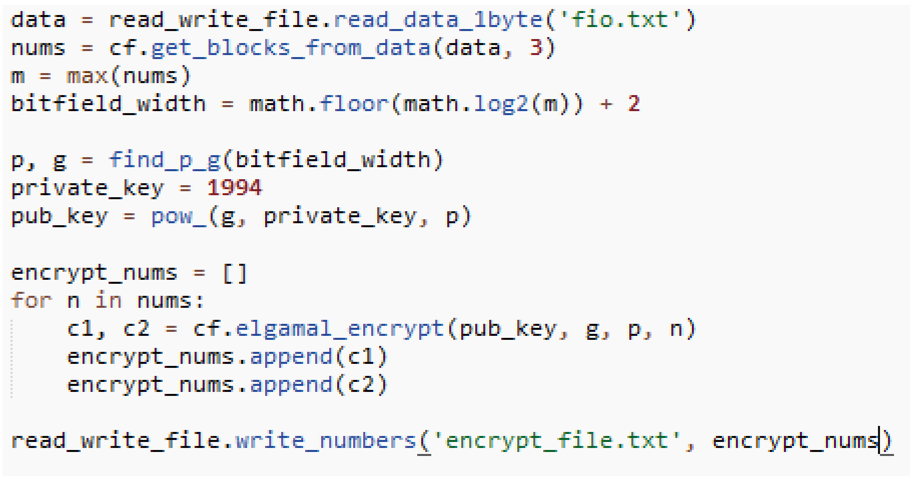
```
import matplotlib.pyplot as plt

bitgr = []
timegr = []
for bit in range(10, 23):
    p, g = find_p_g(bit)
    private_key = p - 3
    pub_key = pow_(g, private_key, p)
    print('p = {}, g = {}, pub_key = {}, private_key = {}'.format(p, g, pub_key, private_key))
    t0 = time.clock()
    private_key = dlog(g, private_key, p)
    t1 = time.clock()

    bitgr.append(bit)
    timegr.append(t1 - t0)
    print(bit, t1 - t0)

plt.plot(bitgr, timegr)
plt.show()
```



Задание 13.

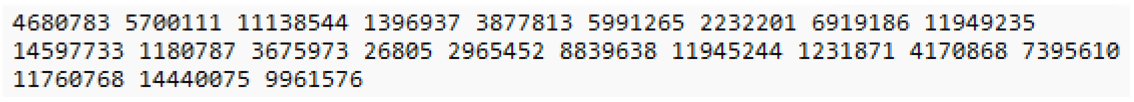
Значения, для которых не существует dlog$_{3,13}$ a: 2, 4, 5, 6, 7, 8, 10, 11, 12.

Задание 14.

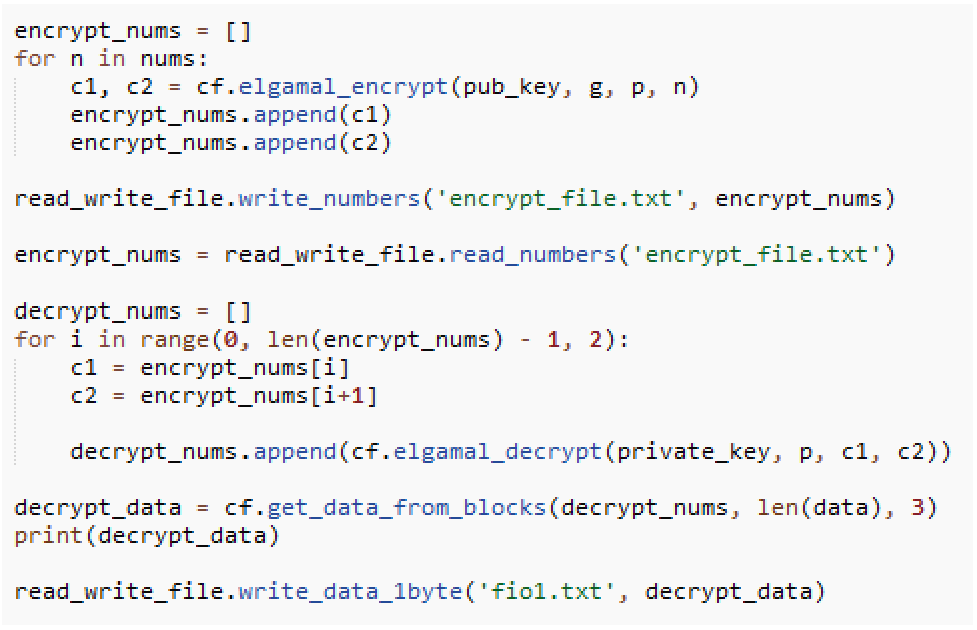


```
data = read_write_file.read_data_1byte('fio.txt')
nums = cf.get_blocks_from_data(data, 3)
m = max(nums)
bitfield_width = math.floor(math.log2(m)) + 2

p, g = find_p_g(bitfield_width)
private_key = 1994
pub_key = pow_(g, private_key, p)

encrypt_nums = []
for n in nums:
    c1, c2 = cf.elgamal_encrypt(pub_key, g, p, n)
    encrypt_nums.append(c1)
    encrypt_nums.append(c2)

read_write_file.write_numbers('encrypt_file.txt', encrypt_nums)
```

Результат:

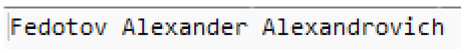


```
4680783 5700111 11138544 1396937 3877813 5991265 2232201 6919186 11949235
14597733 1180787 3675973 26805 2965452 8839638 11945244 1231871 4170868 7395610
11760768 14440075 9961576
```



```
encrypt_nums = []
for n in nums:
    c1, c2 = cf.elgamal_encrypt(pub_key, g, p, n)
    encrypt_nums.append(c1)
    encrypt_nums.append(c2)

read_write_file.write_numbers('encrypt_file.txt', encrypt_nums)

encrypt_nums = read_write_file.read_numbers('encrypt_file.txt')

decrypt_nums = []
for i in range(0, len(encrypt_nums) - 1, 2):
    c1 = encrypt_nums[i]
    c2 = encrypt_nums[i+1]

    decrypt_nums.append(cf.elgamal_decrypt(private_key, p, c1, c2))

decrypt_data = cf.get_data_from_blocks(decrypt_nums, len(data), 3)
print(decrypt_data)

read_write_file.write_data_1byte('fio1.txt', decrypt_data)
```

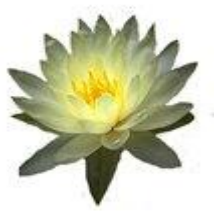
Результат:



```
Fedotov Alexander Alexandrovich
```

Задание 15.

```
data = read_write_file.read_numbers('b4_ElG_c.png')
p = 9887455967
g = 5
pub_key = 3359661584
private_key = 543
block_size = 4

d_nums = []

for i in range(0, len(data) - 1, 2):
    c1 = data[i]
    c2 = data[i+1]

    d_nums.append(cf.elgamal_decrypt(private_key, p, c1, c2))

d_data = cf.get_data_from_blocks(d_nums, len(data), block_size)
read_write_file.write_data_1byte('b4_d.png', d_data)
```



Задание 16.

```
data = read_write_file.read_data_1byte('fio.txt')
nums = cf.get_blocks_from_data(data, 3)
m = max(nums)
bitfield_width = math.floor(math.log2(m)) + 2

p = find_p_2q_plus_1(bitfield_width)
q = find_p_2q_plus_1(bitfield_width)
n = p * q

e = find_p_2q_plus_1(bitfield_width)
while q % e == 1 and p % e == 1:
    e = find_p_2q_plus_1(bitfield_width)

fi_n = (p - 1) * (q - 1)
priv_key = get_inverse_eea(e, fi_n)

ed = []
for d in nums:
    ed.append(rsa_encrypt(d, e, n))

read_write_file.write_numbers('fio_e.txt', ed)

data_e = read_write_file.read_numbers('fio_e.txt')
dd = []
for c in data_e:
    dd.append(rsa_decrypt(c, priv_key, n))

dd = cf.get_data_from_blocks(dd, len(data), 3)
read_write_file.write_data_1byte('fio_d.txt', dd)
```

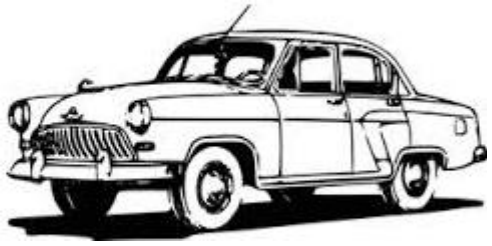
Результат:

```
Fedotov Alexander Alexandrovich
```

Задание 17.

```
data = read_write_file.read_numbers('im49_rsa_c.png')
p = 7919
q = 6599
pub_key = 2011
priv_key = 17457619
len_data = 37451
block_size = 3

d_nums = []

for d in data:
    d_nums.append(rsa_decrypt(d, priv_key, p * q))

d_data = cf.get_data_from_blocks(d_nums, len_data, block_size)
read_write_file.write_data_1byte('im49_rsa_c_decrypt.png', d_data)
```

Результат:



Задание 18.

```
IBECAMEINVOLVEDINANARGUMENTBOUTMODERNPINTINGASUBJECTUPONWHICHIAMSPECT
ACULRLYILLINFORMEDHOWEVERMANYOFMYFRIENDSCANBECOMEHE
TEDANDEVENVIOLENTONTHESUBJECTNDIENJOYTHEIRWRANGLESINMODESTWAYIAMANRTI
STMYSELFNDIHAVESOMESYMPATHYWITHTHEBSTRACTIONISTSLTH
OUGHIHAVEGONEBEYONDTHEMINMYOWNAPPROCHTOARTIMALUMPISTTWOORTHREEDECADES
GOITWSQUITEFASHIONABLETOBEACUBISTANDTODRWEVERYTHING
INCUBESTHENTHEREWASAREVOLTBYTHEVORTICISTSWHODREWEVERYTHINGINWHIRLSWEN
OWHVETHEBSTRACTIONISTSWHOPAINTEVERYTHINGINAVERYABST
RCTEDMNNERBUTMYOWNSMLLWORKSDONEONMYTELEPHONEPADARECOMPOSEDOFCREFULLYS
HADEDSTRANGELYSHAPEDLUMPSWITHTRACESOFCUBISMVORTICIS
MANDABSTRCTIONISMINTHEMFORTHOSEWHOPOSSESSTHESEEINGEYESALUMPISTISTANDA
LONE
```