**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**
**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**
(национальный исследовательский университет)»

Институт №8 «Информационные технологии и прикладная математика»
Кафедра № 806 «Вычислительная математика и программирование»

# Лабораторная работа № 3

По дисциплине «Криптография»

Выполнила студент группы М80-307Б-20:
Сафонникова А. Р.

Принял:
Борисов А. В.

Москва 2023

# ОПИСАНИЕ

Хеш-функция – это функция, которая преобразует произвольный входной набор данных (например, строку, число, файл) в фиксированный размерный набор данных (обычно строку фиксированной длины) - хеш-значение.

Хеш-функции широко используются в информатике для быстрого и эффективного сравнения и проверки целостности данных. Они используются в хеш-таблицах, блокчейнах, криптографии и других областях, где важна быстрая проверка на соответствие.

Хеш-функция должна быть быстрой и эффективной, то есть она должна работать достаточно быстро, даже для больших объемов данных. Кроме того, хеш-функция должна иметь свойство равномерного распределения, что означает, что любые два разных входных значения должны иметь различные хеш-значения.

Хеш-функции могут быть использованы для хранения пар ключ-значение в хеш-таблицах. Ключ может быть произвольным набором данных, а хеш-функция будет использоваться для быстрого поиска соответствующего значения.

Хеш-функции также используются в криптографии для создания электронной подписи и цифровой подписи. В этом случае хеш-функция используется для генерации уникальной строки фиксированной длины для произвольно больших данных, а затем эта строка подписывается с помощью частного ключа, что позволяет убедиться в подлинности и целостности данных.

Хеш-функции также используются в блокчейнах, где они используются для проверки целостности блоков данных, что позволяет убедиться в том, что данные в блоке не были изменены.

# ЗАДАНИЕ

Разложить число на нетривиальные сомножители. В задании представлены 256 вариантов. Вариант выбрать следующим образом: свое ФИО подать на вход в хеш-функцию, являющуюся стандартом, выход хеш-функции представить в шестнадцатеричном виде и рассматривать младший байт как номер варианта. В отчете привести подробности процесса вычисления номера варианта.

# РЕШЕНИЕ

Для того, чтобы определить вариант задания, я использовала стандартную библиотеку *hashlib* языка программирования python. Таким образом был определён вариант под цифрой 108.

```
import hashlib
```

получаем хеш в шестандцетиричном виде

```
def calculate_hash(FIO):
    hash_object = hashlib.sha256(FIO.encode())
    hex_dig = hash_object.hexdigest()
    return hex_dig
```

```
hash_value = calculate_hash("Сафонникова Анна Романовна")
print("Хэш-код полученного значения:", hash_value)
```

```
Хэш-код полученного значения: 43f165284f9dc615fc4c366230f78ec6f9ce0e960f15264df7da4b9a9491426c
```

преобразуем hex-строку в байты с помощью библиотеки codecs и получим последний байт

```
import codecs
```

```
bytes_obj = codecs.decode(hex_dig, "hex")

last_byte = bytes_obj[-1]
print("Вариант лабораторной работы:", last_byte)
```

```
Вариант лабораторной работы: 108
```

Теперь разложим число на нетривиальные множители. Факторизация оказалась слишком долгим способом, поэтому первый множитель найдём как НОД с числом другого варианта,  а второй делением.

Теперь напишем функцию факторизации:

```
def prime_factors(n):
    factors = []
    for i in range(2, n):
        if gcd(n, i) == i:
            factors.append(i)
            while n % i == 0:
                n //= i
    if n > 1:
        factors.append(n)
    return factors


num = 3894781440679692957390691832671971149486946707408697627519984801086192823638545488585834231577625769465891914723552055

factors = prime_factors(num)
print(factors)
```

Программа работает слишком долго, и я даже не уверена, что она справится с 309-значным числом..

Изучив методы решения других, я наткнулась на некую подсказку, что среди чисел других вариантов есть то число, которое имеет общее НОД с моим. Что ж, это сильно облегчает задачу, так как благодаря этому мы можем найти первый множитель, а второй с помощью обычного деления:

```
num = 3894781440679692957390691832671971149486946707408697627519984801086192823638545488585834231577625769465891914723552055059302950855773554213025317946457997813388086820690427263451716947706684848756607004402035045032903289117004983802288725863825865014574013889902275945861966814641367445622824954789855985729
numbers = [
  2890245022085769479617158032473283620929696309984324448388971308342808198848841583217610751958275960712898246538792067197142410318043885920428874982971388265266396963240516124923505341114226965890545463880062256947046838457308458913854886082596310391298031096313745784619574622660174584937828372712457553575337680912629702887436233106791538175219829975317660065232874384082460666464897506402714346030456563163519971457853532576426530164479686688152563037167470915034354305782904216892851165178496366610231584395766491230012103377962610967055925356215794030550406522251836985912958786786116329479725084660699288780514885559862831976804400
]

i = 0
while find_gcd(num, numbers[i]) == 1:
  i += 1

num1 = find_gcd(num, numbers[i])
num2 = num // num1

print("Нетривиальные множители:")
print("num1 = ", num1)
print("num2 = ", num2)
```

```
Нетривиальные множители:
num1 =  167330276945913172339257510585316077547136560268409878494186502768429972781653
num2 =  232760114413640669276440659212883279996301329970904568498879547432268675773551701887198132652524222196713771636009232878034683726578381815072819932594299463789913088801512722613777680043653412008371995736007773349860784337782389178
```

В итоге было получено:

Число:

389478144067969295739069183267197114948694670740869762751998480108619282363854588585834231577625769465891914723552055059302950855773554213025317946457997813388086820690427263451716947706684848756607004402035045032903289117004983802288725863825865014574013889902275945861966814641367445622824954789855985729

Нетривиальные множители:

1) 167330276945913172339257510585316077547136560268409878494186502768429972781653

2) 232760114413640669276440659212883279996301329970904568498879547432268675773551701887198132652524222196713771636009232878034683726578381815072819932594299463789913088801512722613777680043653412008371995736007773349860784337782389178

# ВЫВОД

В ходе выполнения лабораторной работы был использован метод разложения числа на нетривиальные сомножители. Для выбора варианта было использовано хеширование ФИО с помощью стандартной хеш-функции, а затем приведение результата в шестнадцатеричную систему счисления и выбор младшего байта как номера варианта.