**Name : Akash Babu Misal**

**PRN: 21520005**

**Class:** Final Year (Computer Science and Engineering)

**Course Name: Cryptography and Network Security**  **Lab**

**Assignment No – 7**

**Aim**:- Prime Factorization of large numbers

**Theory**: We have to factorize a number such that its factors are prime and their product equals a given number.

**Code**:

# !/usr/bin/env python

from sympy import \*

from random import randint

def miller\_rabin(n, k):

    if n == 2 or n == 3:

        return True

    if n < 2 or n % 2 == 0:

        return False

    r, s = 0, n - 1

    while s % 2 == 0:

        r += 1

        s //= 2

    for \_ in range(k):

        a = randint(2, n - 2)

        x = pow(a, s, n)

        if x == 1 or x == n - 1:

            continue

        for \_ in range(r - 1):

            x = pow(x, 2, n)

            if x == n - 1:

                break

        else:

            return False

    return True

def factorize(n):

    k = int(log(n, 2))

    while True:

        p = randprime(10\*\*(k-1), 10\*\*k)

        q = randprime(10\*\*(k-1), 10\*\*k)

        if p\*q >= n:

            break

    return p, q

n = randprime(10\*\*49, 10\*\*50)

p, q = factorize(n)

print(f"n: {n}")

print(f"p: {p}")

print(f"q: {q}")

**Output –**

**A black background with many small colored lines

Description automatically generated with medium confidence**

**Conclusion**:

The prime factorization of large numbers is a fascinating mathematical and computational problem. It has practical applications in cryptography, where large numbers are used for encryption and decryption, and in number theory. This experiment helps to explore and understand different methods for prime factorization and how they can be used to break down large numbers into their prime components. It also highlights the importance of efficient algorithms for dealing with very large numbers.

**Aim**: Find the GCD of two given number using Euclidean Algo

**Theory**:

**Greatest Common Divisor (GCD) Overview:**

The GCD of two numbers, often denoted as GCD(a, b), is the largest positive integer that divides both numbers without leaving a remainder. Calculating the GCD of two numbers is a fundamental operation in number theory and has various applications in mathematics and computer science.

**Euclidean Algorithm:**

The Euclidean Algorithm is a widely used method for finding the GCD of two numbers. It is based on the principle that the GCD of two numbers remains the same if the smaller number is subtracted from the larger one. The algorithm iteratively applies this subtraction until the two numbers are equal, at which point the GCD is found.

**Algorithm Steps:**

The Euclidean Algorithm to find the GCD of two numbers 'a' and 'b' involves the following steps:

1. Start with the two given numbers, 'a' and 'b'.
2. If 'b' is 0, then the GCD is 'a', so return 'a' as the result.
3. Otherwise, calculate the remainder of 'a' divided by 'b', denoted as 'r'

(r = a % b).

1. Replace 'a' with 'b' and 'b' with 'r'.
2. Repeat steps 2 to 4 until 'b' becomes 0.
3. The GCD is the remaining non-zero value of 'a'.

**Code**:

#include <bits/stdc++.h>

using namespace std;

void file()

{

#ifndef ONLINE\_JUDGE

    freopen("input.txt", "r", stdin);

    freopen("output.txt", "w", stdout);

#endif

}

int findGCD(int num1, int num2)

{

    if (num2 == 0)

        return num1;

    cout << setprecision(10) << num1 / num2 << "\t" << setprecision(10) << num1 << "\t" << setprecision(10) << num2 << "\t" << setprecision(10) << num1 % num2 << endl;

    return findGCD(num2, num1 % num2);

}

int main()

{

    file();

    int num1, num2;

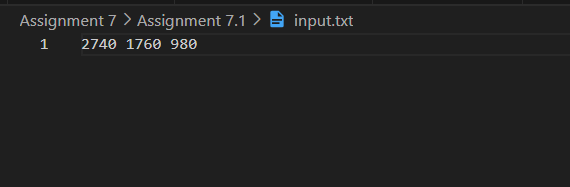
    cin >> num1 >> num2;

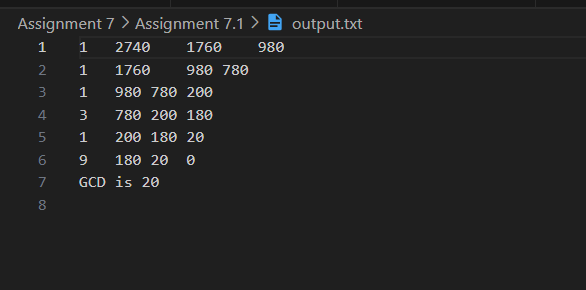
    int gcd = findGCD(num1, num2);

    cout << "GCD is " << gcd << endl;

    return 0;

}





**Conclusion:**

The Euclidean Algorithm is an efficient and well-established method for finding the GCD of two numbers. This experiment demonstrates the use of the algorithm to determine the GCD and highlights its significance in various mathematical and computational applications, including simplifying fractions, determining the highest common factors, and solving problems related to number theory and modular arithmetic.