**Name : Akash Babu Misal**

**PRN: 21520005**

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

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

**Assignment No – 8**

**Aim**:

Find the GCD of two given numbers using Extended Euclidean Algorithm

**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 is a fundamental operation in number theory and has various applications in mathematics and computer science.

**Extended Euclidean Algorithm:**

The Extended Euclidean Algorithm is an extension of the standard Euclidean Algorithm. While it calculates the GCD of two numbers, it also finds Bézout coefficients (x and y) for the linear Diophantine equation ax + by = GCD(a, b). This makes it valuable for solving problems related to modular arithmetic, cryptography, and number theory.

**Algorithm Steps:**

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

1. Start with the two given numbers, 'a' and 'b'.
2. Initialize the coefficients x1, x2, y1, y2, x, and y as follows:

x1 = 1, x2 = 0, y1 = 0, y2 = 1.

1. Perform the standard Euclidean Algorithm to find the GCD of 'a' and 'b'.
2. During the process, maintain and update x and y as follows:

* x = x1 - (a // b) \* x2
* y = y1 - (a // b) \* y2

1. Continue the Euclidean Algorithm until 'b' becomes 0.
2. The GCD is the remaining non-zero value of 'a'.
3. The coefficients x and y are the Bézout coefficients for the linear Diophantine equation ax + by = GCD(a, b).

A table with numbers and equations

Description automatically generated

**Code:**

#include <bits/stdc++.h>

using namespace std;

int ansS, ansT;

int findGcdExtended(int r1, int r2, int s1, int s2, int t1, int t2)

{

    // Base Case

    if (r2 == 0)

    {

        ansS = s1;

        ansT = t1;

        return r1;

    }

    int q = r1 / r2;

    int r = r1 % r2;

    int s = s1 - q \* s2;

    int t = t1 - q \* t2;

    cout << q << " " << r1 << " " << r2 << " " << r << " " << s1 << " " << s2 << " " << s << " " << t1 << " " << t2 << " " << t << endl;

    return findGcdExtended(r2, r, s2, s, t2, t);

}

int main()

{

    int num1, num2;

    cout << "\n Enter 1st number : ";

    cin >> num1;

    cout << "\n Enter 2nd number : ";

    cin >> num2;

    cout << endl

         << "q r1 r2 r s1 s2 s t1 t2 t" << endl;

    int gcd = findGcdExtended(num1, num2, 1, 0, 0, 1);

    cout << endl

         << "GCD is " << gcd << endl;

    cout << endl

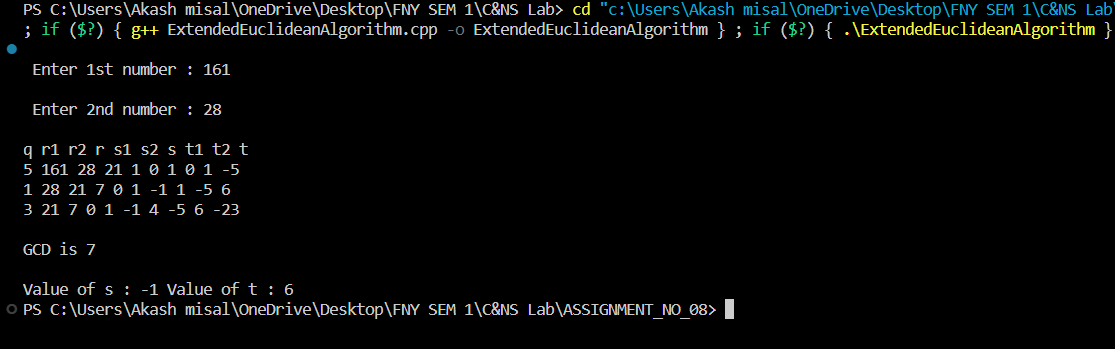
         << "Value of s : " << ansS << " "

         << "Value of t : " << ansT << endl;

    return 0;

}

**Output**:



**Conclusion**:

The Extended Euclidean Algorithm is a powerful tool for finding the GCD of two numbers and obtaining the Bézout coefficients for a linear Diophantine equation. This experiment demonstrates the use of the algorithm to determine the GCD and the coefficients, highlighting its importance in modular arithmetic, cryptography, and problem-solving in number theory. The Bézout coefficients can be particularly valuable in solving problems where finding solutions to linear equations with integer constraints is necessary.