**Final Year B. Tech., Sem VII 2022-23**

**Cryptography And Network Security**

**PRN/ Roll No: 2020BTECS00206**

**Full Name: SAYALI YOGESH DESAI**

**Batch: B4**

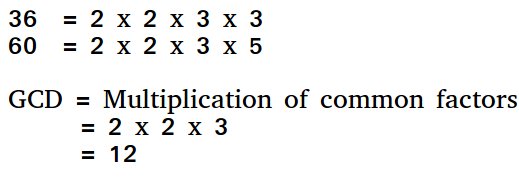
**Assignment No. 8**

1. **Aim:**

Implementation of Euclidean and Extended Euclidean Algorithm.

1. **Theory:**

The Euclidean algorithm is a way to find the greatest common divisor of two positive integers. GCD of two numbers is the largest number that divides both of them. A simple way to find GCD is to factorize both numbers and multiply common prime factors.



**Basic Euclidean Algorithm for GCD:**

The algorithm is based on the below facts.

* If we subtract a smaller number from a larger one (we reduce a larger number), GCD doesn’t change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.
* Now instead of subtraction, if we divide the smaller number, the algorithm stops when we find the remainder 0.

Extended Euclidean algorithm also finds integer coefficients x and y such that: ax + by = gcd(a, b)

**Examples:**

***Input:****a = 30, b = 20****Output:****gcd = 10, x = 1, y = -1  
(Note that 30\*1 + 20\*(-1) = 10)*

***Input:****a = 35, b = 15****Output:****gcd = 5, x = 1, y = -2  
(Note that 35\*1 + 15\*(-2) = 5)*

## ****How does Extended Algorithm Work?****

As seen above, x and y are results for inputs a and b,

*a.x + b.y = gcd                      —-(1)*

And x1 and y1 are results for inputs b%a and a*(b%a).x1 + a.y1 = gcd*

When we put b%a = (b – (⌊b/a⌋).a) in above, we get following. Note that ⌊b/a⌋ is floor(b/a) *(b – (⌊b/a⌋).a).x1 + a.y1  = gcd*

Above equation can also be written as below

*b.x1 + a.(y1 – (⌊b/a⌋).x1) = gcd      —(2)*

After comparing coefficients of ‘a’ and ‘b’ in (1) and (2), we get following,   
*x = y1 – ⌊b/a⌋ \* x1  
y = x1*

## ****How is Extended Algorithm Useful?****

The extended Euclidean algorithm is particularly useful when a and b are coprime (or gcd is 1). Since x is the modular multiplicative inverse of “a modulo b”, and y is the modular multiplicative inverse of “b modulo a”. In particular, the computation of the modular multiplicative inverse is an essential step in RSA public-key encryption method.

* **Euclidean Algorithm:**

#include <bits/stdc++.h>

using namespace std;

int findGCD(int num1, int num2)

{

if (num1 == 0)

return num2;

return findGCD(num2 % num1, num1);

}

int main()

{

int num1, num2;

cout << "\n Enter 1st number : ";

cin >> num1;

cout << "\n Enter 2nd number : ";

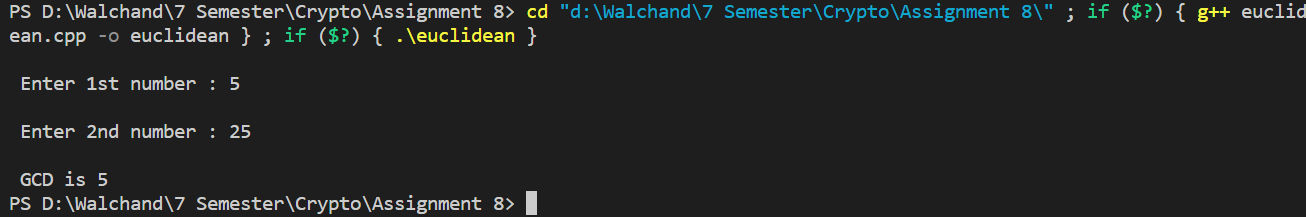
cin >> num2;

int gcd = findGCD(num1, num2);

cout << "\n GCD is " << gcd << endl;

return 0;

}

****

* **Extended Euclidean Algorithm:**

#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, s, t;

cout << "\n Enter 1st number : ";

cin >> num1;

cout << "\n Enter 2nd number : ";

cin >> num2;

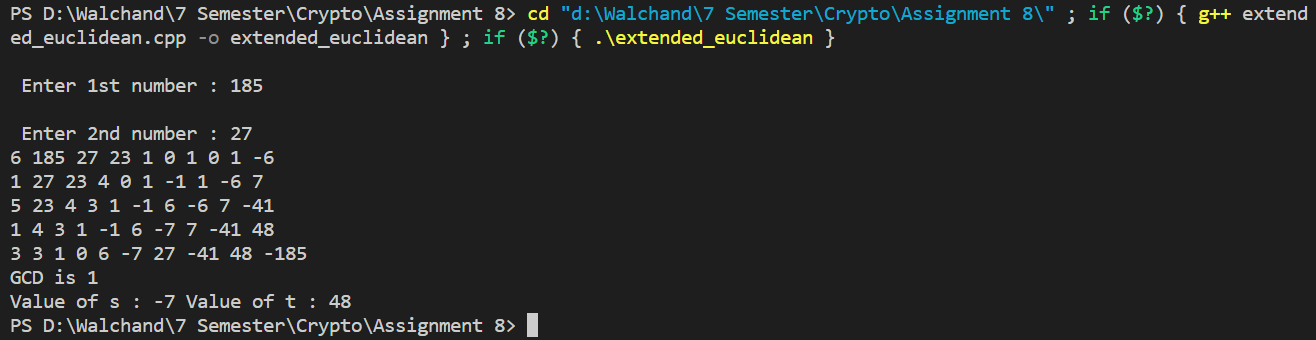
int gcd = findGcdExtended(num1, num2, 1, 0, 0, 1);

cout << "GCD is " << gcd << endl;

cout << "Value of s : "<<ansS << " " <<"Value of t : "<<ansT << endl;

return 0;

}



1. **Conclusion:**

The Euclidean and Extended Euclidean algorithm are used to find the GCD of numbers and the Multiplicative inverse of two coprime numbers respectively.