

Cryptography And Network Security Lab

Assignment submission

PRN No: 2019BTECS00017

Full name: Muskan Raju Attar

Batch: B5

Assignment: 8

**Title of assignment: Implementation of Euclidean and
Extended Euclidean Algorithm**

Title:

Implementation of Euclidean and Extended Euclidean Algorithm

Aim:

To develop and implement the Euclidean and Extended Euclidean Algorithm

Theory:

- In mathematics, the Euclidean algorithm, or Euclid's algorithm, is an efficient method for computing the greatest common divisor (GCD) of two integers (numbers), the largest number that divides them both without a remainder.
- The Euclidean algorithm is based on the principle that the greatest common divisor of two numbers does not change if the larger number is replaced by its difference with the smaller number. For example, 21 is the GCD of 252 and 105 (as $252 = 21 \times 12$ and $105 = 21 \times 5$), and the same number 21 is also the GCD of 105 and $252 - 105 = 147$.

- By using the extended Euclidean algorithm, the GCD can be expressed as a linear combination of the two original numbers, that is the sum of the two numbers, each multiplied by an integer (for example, $21 = 5 \times 105 + (-2) \times 252$). The fact that the GCD can always be expressed in this way is known as Bézout's identity.
- The Euclidean algorithm has many theoretical and practical applications. It is used for reducing fractions to their simplest form and for performing division in modular arithmetic. Computations using this algorithm form part of the cryptographic protocols that are used to secure internet communications, and in methods for breaking these cryptosystems by factoring large composite numbers.
- In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the greatest common divisor (gcd) of integers a and b , also the coefficients of Bézout's identity, which are integers x and y such that $ax + by = \gcd(a, b)$.
- The extended Euclidean algorithm is particularly useful when a and b are coprime. With that provision, x is the modular multiplicative inverse of a modulo b , and y is the modular multiplicative inverse of b modulo a .

Implementation of Euclidean Algorithm

Code:

```
#include<bits/stdc++.h>
```

```

using namespace std;
typedef long long int ll;

// function to find gcd of two integer numbers
ll gcd(ll a, ll b)
{
    if (!a)
        return b;
    return gcd(b % a, a);
}

ll reduceB(ll a, char b[])
{
    // Initialize result
    ll mod = 0;

    // calculating mod of b with a to make
    // b like 0 <= b < a
    for (int i = 0; i < strlen(b); i++)
        mod = (mod * 10 + b[i] - '0') % a;

    return mod; // return modulo
}

ll gcdLarge(ll a, char b[])
{
    // Reduce 'b' (second number) after modulo with a
    ll num = reduceB(a, b);

    // gcd of two numbers
    return gcd(a, num);
}

int main()
{

```

```

// first number which is integer
ll a = 1221;

char b[] =
"1234567891011121314151617181920212223242526272829";

cout<<"Enter a Smaller Number: ";
cin>>a;

cout<<"Enter a Large Number: ";
cin>>b;

cout<<"\nThe GCD of Given Number is: ";

if (a == 0)
    cout << b << endl;
else
    cout << gcdLarge(a, b) << endl;

return 0;
}

```

Output:

C:\Users\Muskan Raju Attar\Desktop\CNS Assignment new\Experiment - 8\Euclidean.exe

Enter a Smaller Number: 35

Enter a Large Number: 56

The GCD of Given Number is: 7

Process exited after 6.858 seconds with return value 0
Press any key to continue . . .

Implementation of Extended Euclidean Algorithm

Code:

```
#include<bits/stdc++.h>
```

```
typedef long long LL;
```

```
void extended_Euclidean_algorithm(LL a, LL b, LL &u, LL &v, LL &w, LL  
&x, LL &y, LL &z){
```

```
    /* Initialization */
```

```
    // 1. equation
```

```
    u = 1; v = 0; w = a;
```

```
    // 2. equation
```

```
    x = 0; y = 1; z = b;
```

```
    if( w < z ){ // we change the equations' order
```

```
        std::swap( u, x );
```

```
        std::swap( v, y );
```

```
        std::swap( w, z );
```

```
    }
```

```
    LL q;
```

```
    while( z != 0 ){
```

```
        q = w / z;
```

```
        // (1. equation) - q * (2. equation)
```

```
        u -= q*x;
```

```
        v -= q*y;
```

```
        w -= q*z;
```

```
        // we change the equations' order
```

```
        std::swap( u, x );
```

```
        std::swap( v, y );
```

```
        std::swap( w, z );
```

```
    }  
}
```

```
int main(){
```

```
    LL a, b, u, v, w, x, y, z;
```

```
    // -----
```

```
    printf( "Data input\n" );
```

```
    printf( "a = " );
```

```
    scanf( "%lld", &a );
```

```
    printf( "b = " );
```

```
    scanf( "%lld", &b );
```

```
    // -----
```

```
    extended_Euclidean_algorithm(a, b, u, v, w, x, y, z);
```

```
    // -----
```

```
    printf( "\nResults:\n" );
```

```
    printf( "1. equation: (%lld)*(%lld) + (%lld)*(%lld) = %lld\n", a, u, b,  
v, w );
```


```
    printf( "2. equation: (%lld)*(%lld) + (%lld)*(%lld) = %lld\n", a, x, b,  
y, z );
```

```
    // -----
```

```
    return 0;
```

```
}
```

Output:

 C:\Users\Muskan Raju Attar\Desktop\CNS Assignment new\Experiment - 8\Extended_Euclidean.exe

Data input

a = 54


b = 34

Results:

1. equation: $(54)*(-5) + (34)*(8) = 2$

2. equation: $(54)*(17) + (34)*(-27) = 0$

Process exited after 11.7 seconds with return value 0

Press any key to continue . . . 

Conclusion:

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