**Experiment No.1**

**Aim:** Write a program to implement Extended Euclidean Algorithm using C++ or JAVA

**Objectives:** To study

* Extended Euclidean Algorithm
* To calculate GCD using Extended Euclidean Algorithm
* How to use in RSA Extended Euclidean Algorithm

**Theory:**

**Explanation:**

The *Extended Euclidean Algorithm* is just a fancier way of Using the Euclidean algorithm above. It involves using extra variables to compute ax + by = gcd(a, b) as we go through the Euclidean algorithm in a single pass. It's more efficient to use in a computer program.

### **Euclidean algorithm**

The *Euclidean algorithm* is an efficient method to compute the *greatest common divisor* (gcd) of two integers. It was first published in Book VII of Euclid's *Elements* sometime around 300 BC.

We write gcd(a, b) = d to mean that *d* is the largest number that will divide both *a* and *b*. If gcd(a, b) = 1 then we say that *a* and *b* are *co-prime* or *relatively prime*. The gcd is sometimes called the *highest common factor* (hcf).

**Algorithm:** (Euclidean algorithm) Computing the greatest common divisor of two integers.

**INPUT:** Two non-negative integers *a* and *b* with a ≥ b.

**OUTPUT: gcd(a, b).**

1. While b > 0, do
   1. Set r = a mod b,
   2. a = b,
   3. b = r
2. Return *a*.

**Question :** Find gcd(421, 111).

**Answer:** We use the Euclidean algorithm as follows:

|  |  |
| --- | --- |
| 421 = 111 x 3 + 88 | (larger number on left) |
| 111 = 88 x 1 + 23 | (shift left) |
| 88 = 23 x 3 + 19 | (note how 19 moves down the "diagonal") |
| 23 = 19 x 1 + 4 |  |
| 19 = 4 x 4 + 3 |  |
| 4 = 3 x 1 + **1** | (last non-zero remainder is 1) |
| 3 = 1 x 3 + 0 |  |

The last non-zero remainder is 1 and therefore gcd(421, 111) = 1.

**The Extended Euclidean Algorithm**

The *Extended Euclidean Algorithm* is just a fancier way of doing what we did [Using the Euclidean algorithm](http://www.di-mgt.com.au/euclidean.html#usingeuclidean) above. It involves using extra variables to compute ax + by = gcd(a, b) as we go through the Euclidean algorithm in a single pass. It's more efficient to use in a computer program.

Shape

**Algorithm:** Extended Euclidean algorithm.

Shape

**INPUT:** Two non-negative integers *a* and *b* with a ≥ b.  
**OUTPUT:** d = gcd(a, b) and integers *x* and *y* satisfying ax + by = d.

1. If b = 0 then set d = a, x = 1, y = 0, and return(*d, x, y*).
2. Set x2 = 1, x1 = 0, y2 = 0, y1 = 1
3. While b > 0, do
   1. q = floor(a/b), r = a - qb, x = x2 - qx1, y = y2 - q y1.
   2. a = b, b = r, x2 = x1, x1 = x, y2 = y1, y1 = y.
4. Set d = a, x = x2, y = y2, and return(*d, x, y*).

with input *a* = 4864, *b* = 3458 we get following values

----- --------------------------------------------

q r x y a b x2 x1 y2 y1

-------------------------------------------------

1 1406 1 -1 3458 1406 0 1 1 -1

2 646 -2 3 1406 646 1 -2 -1 3

2 114 5 -7 646 114 -2 5 3 -7

5 76 -27 38 114 76 5 -27 -7 38

1 38 32 -45 76 38 -27 32 38 -45

2 0 -91 128 38 0 32 -91 -45 128

-------------------------------------------------

x = 32 y = -45 d = 38

**That is, gcd(4864, 3458) = 38 and 32 x 4864 - 45 x 3458 = 38.**

**INPUT:** Values of a and b

**OUTPUT: V**alue of x,y and GCD

**Conclusion**: We have successfully implemented Extended Euclidean Algorithm in C++.

**Source Code**:

**To Implement Extended Euclidian Algorithm using C++ or JAVA**

#include<bits/stdc++.h>

using namespace std;

// Function for extended Euclidean Algorithm

int gcdExtended(int a, int b, int \*x, int \*y)

{

if (a == 0)

{

\*x = 0;

\*y = 1;

return b;

}

int x1, y1;

int gcd = gcdExtended(b%a, a, &x1, &y1);

\*x = y1 - (b/a) \* x1;

\*y = x1;

return gcd;

}

int main()

{

int x, y, num1 , num2;

cout<<"Enter the First number : ";

cin>>num1;

cout<<"Enter the Second number : ";

cin>>num2;

int g = gcdExtended(num1, num2, &x, &y);

cout << "GCD(" << num1 << ", " << num2 << ") = " << g << endl;

cout << "Coefficient are : "<< x << " " << y <<endl;

return 0;

}

**Snapshot :**

