Final Year B. Tech. (CSE) – I: 2022-23

4CS451: Cryptography and Network Security Lab

Assignment No. 8

PRN: 2019BTECS00077 **Batch: B7**

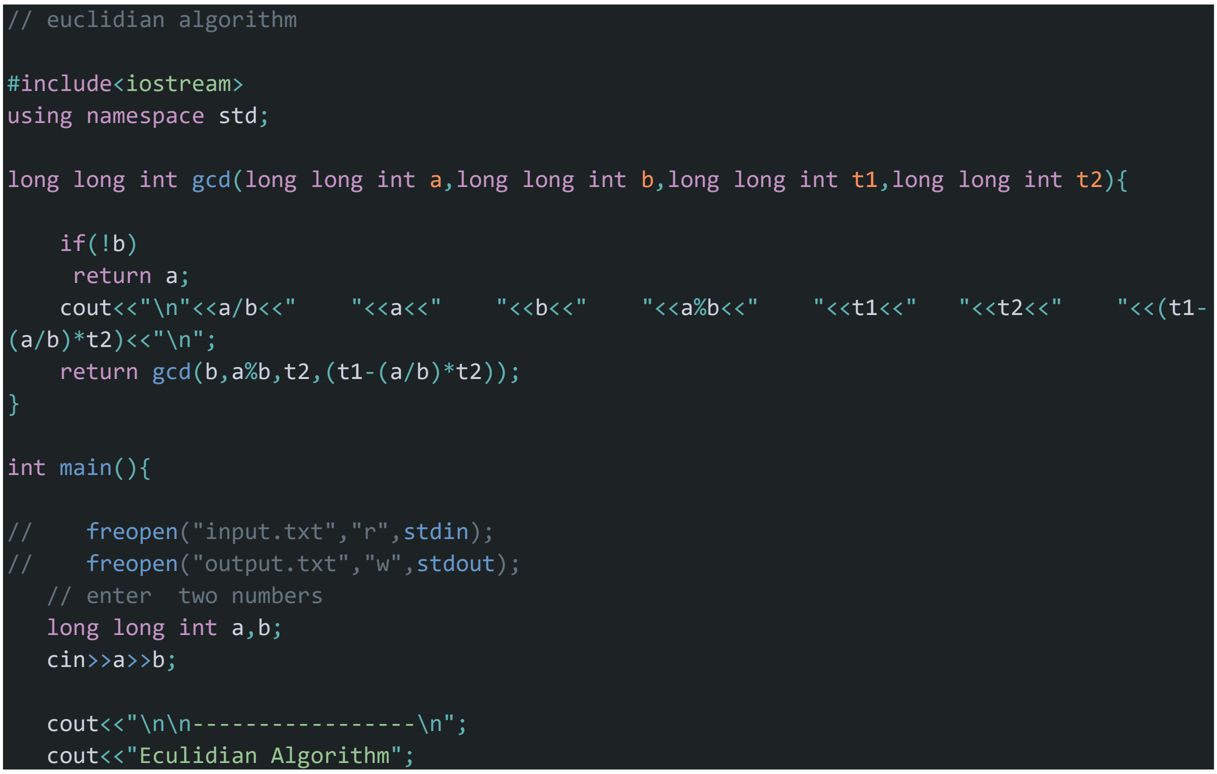
Full name: Biradar Avinash Vishnu

Title: Euclidian and Extended Euclidian Algorithm.

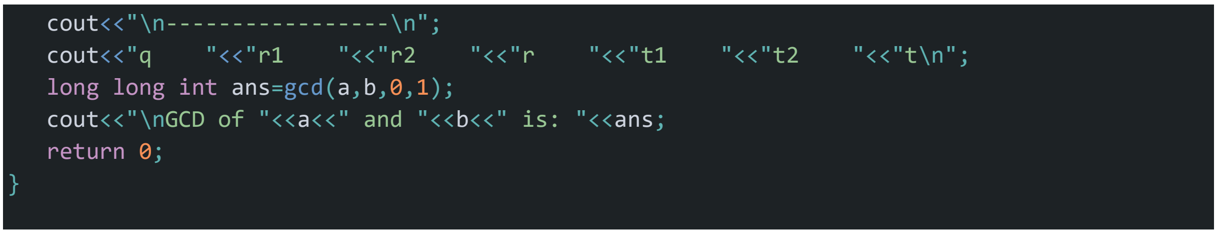
Objective: To find gcd using eculidian and multiplicative invese using extended eculidian algorithm.

1. Eculidian Algorithm implementation and output:

Code:

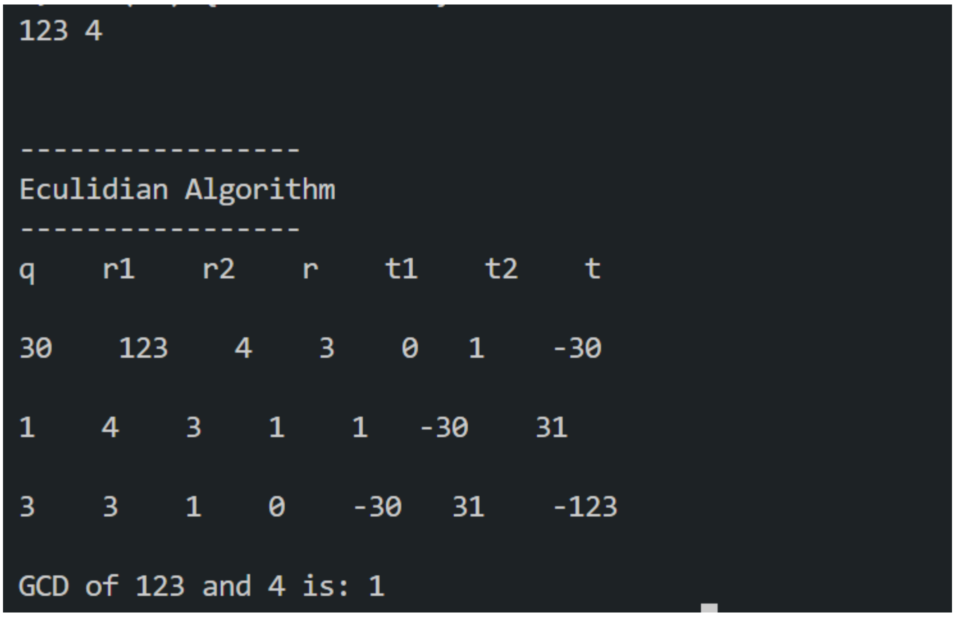


```
// euclidian algorithm

#include<iostream>
using namespace std;

long long int gcd(long long int a,long long int b,long long int t1,long long int t2){

    if(!b)
     return a;
    cout<<"\n"<<a/b<<"    "<<a<<"    "<<b<<"    "<<a%b<<"    "<<t1<<"    "<<t2<<"    "<<(t1-
(a/b)*t2)<<"\n";
    return gcd(b,a%b,t2,(t1-(a/b)*t2));
}

int main(){

//    freopen("input.txt","r",stdin);
//    freopen("output.txt","w",stdout);
   // enter  two numbers
   long long int a,b;
   cin>>a>>b;

   cout<<"\n\n-----------------\n";
   cout<<"Eculidian Algorithm";
```

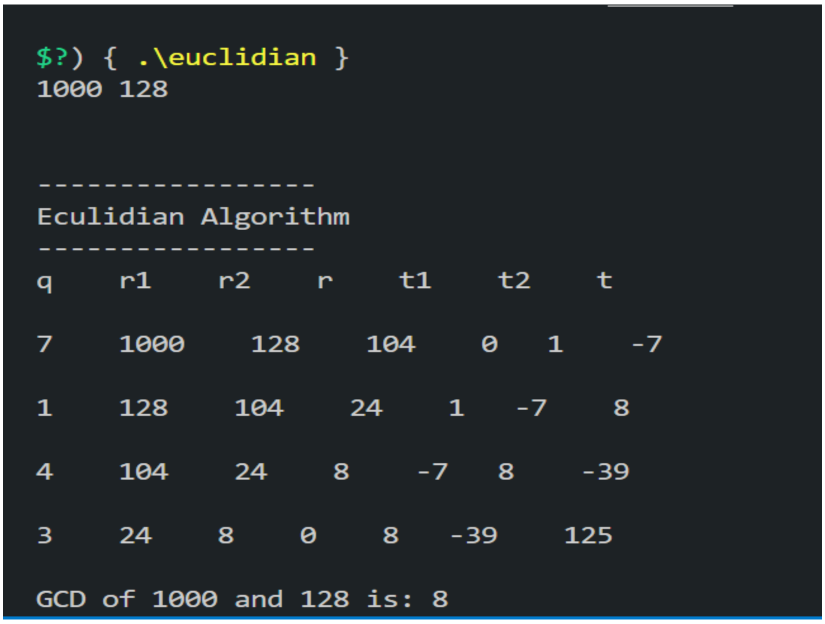


```
    cout<<"\n-----------------\n";
    cout<<"q     "<<"r1     "<<"r2     "<<"r     "<<"t1     "<<"t2     "<<"t\n";
    long long int ans=gcd(a,b,0,1);
    cout<<"\nGCD of "<<a<<" and "<<b<<" is: "<<ans;
    return 0;
}
```

Output:



```
123 4


-----------------
Eculidian Algorithm
-----------------
q     r1     r2     r     t1     t2     t

30     123     4     3     0    1     -30

1     4     3     1     1    -30     31

3     3     1     0     -30    31     -123

GCD of 123 and 4 is: 1
```



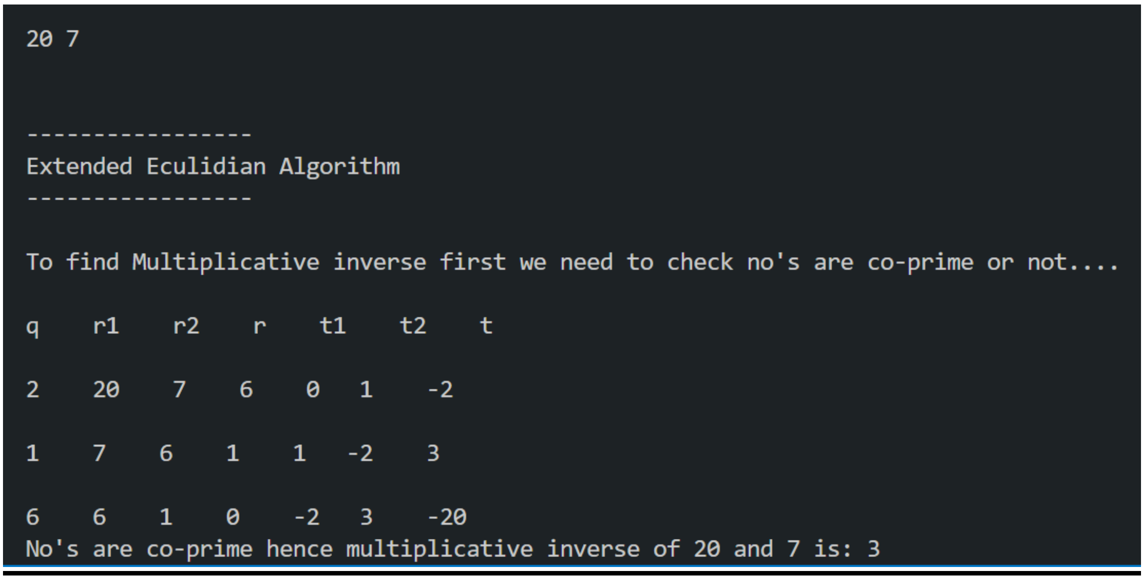
```
$?) { .\euclidian }
1000 128


-----------------
Eculidian Algorithm
-----------------
q     r1     r2     r     t1     t2     t

7     1000     128     104     0    1     -7

1     128     104     24     1    -7     8

4     104     24     8     -7    8     -39

3     24     8     0     8    -39     125

GCD of 1000 and 128 is: 8
```

2. Extended Eculidian implementation and output:

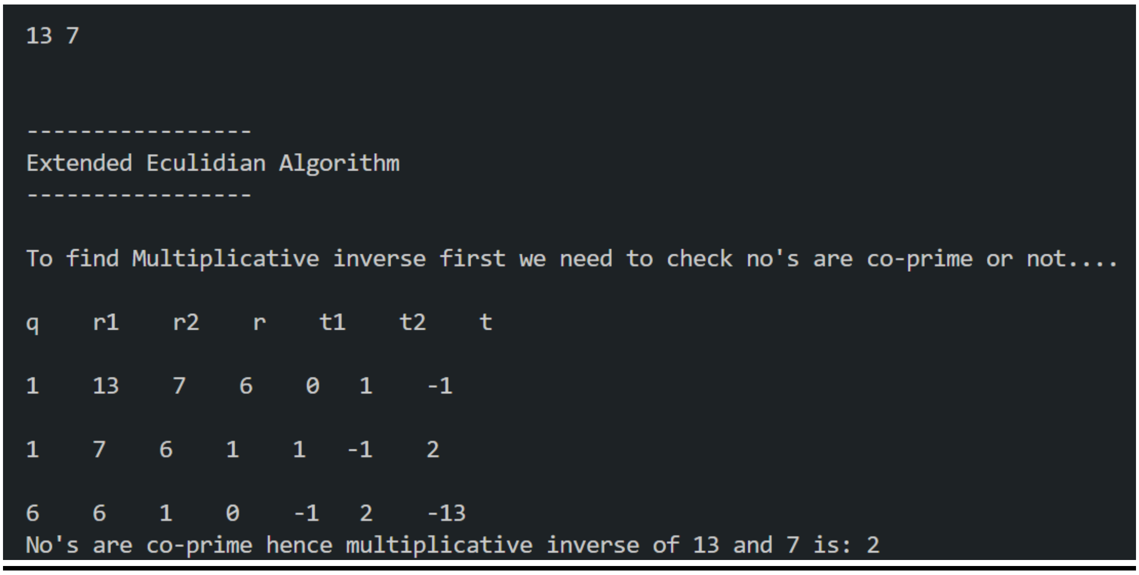
Code:

```
// euclidian algorithm

#include<iostream>
using namespace std;

long long int gcd(long long int a,long long int b,long long int &t1,long long int &t2){

    if(!b)
     return a;
    cout<<"\n"<<a/b<<"    "<<a<<"    "<<b<<"    "<<a%b<<"    "<<t1<<"    "<<t2<<"    "<<(t1-(a/b)*t2)<<"\n";
    t1=(t1-(a/b)*t2);
    return gcd(b,a%b,t2,t1);
}

int main(){

//    freopen("input.txt","r",stdin);
//    freopen("output.txt","w",stdout);
   // enter  two numbers
   long long int a,b;
   cin>>a>>b;

   cout<<"\n\n-----------------\n";
   cout<<"Extended Eculidian Algorithm";
   cout<<"\n-----------------\n";

   cout<<"\nTo find Multiplicative inverse first we need to check no's are co-prime or not....\n\n";
   cout<<"q    "<<"r1    "<<"r2    "<<"r    "<<"t1    "<<"t2    "<<"t\n";
   long long int t1=0,t2=1;
   long long int ans=gcd(a,b,t1,t2);
   if(ans!=1){
      cout<<"No's are Not Co-prime...end...";
      return 0;
   }
   cout<<"No's are co-prime hence multiplicative inverse of "<<a<<" and "<<b<<" is: "<<t2;
   return 0;
```

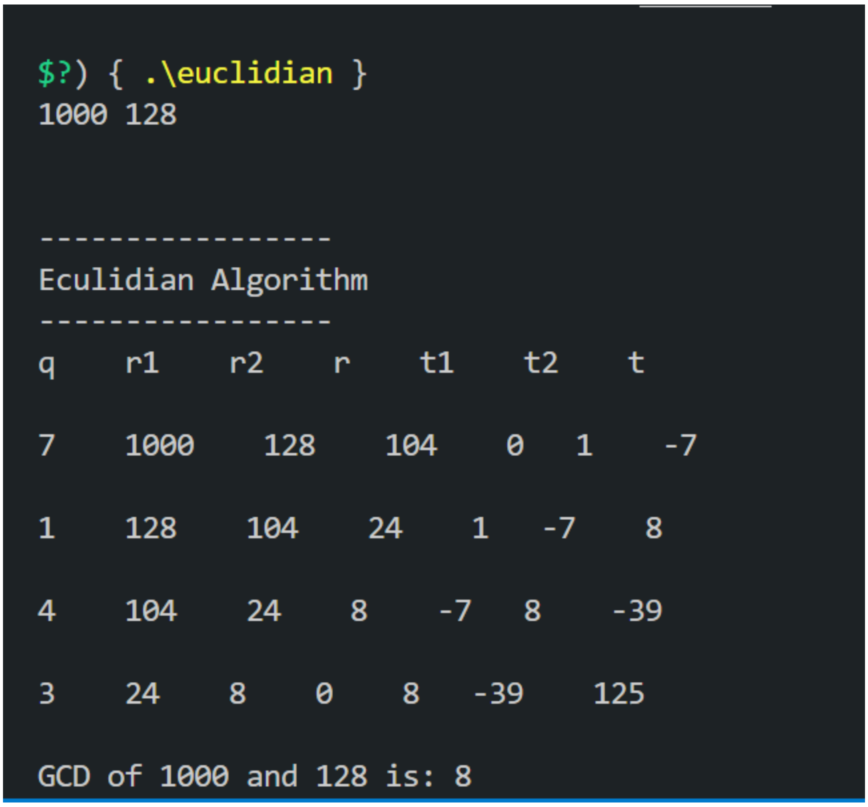
Output:



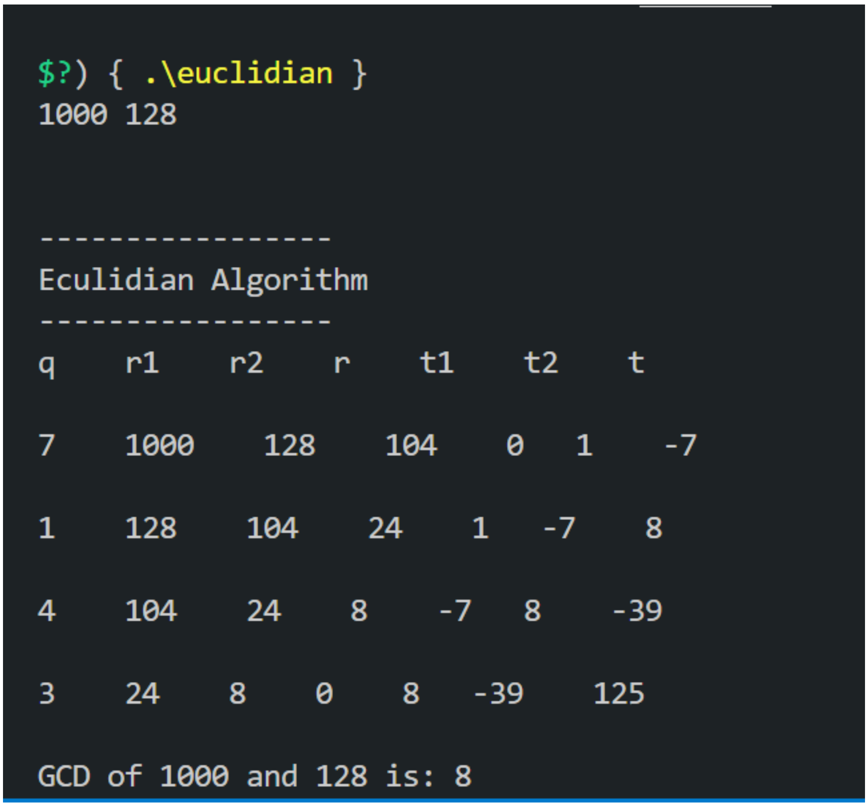
```
20 7


-----------------
Extended Eculidian Algorithm
-----------------

To find Multiplicative inverse first we need to check no's are co-prime or not....

q    r1    r2    r    t1    t2    t

2    20    7    6    0   1    -2

1    7    6    1    1   -2    3

6    6    1    0    -2   3    -20
No's are co-prime hence multiplicative inverse of 20 and 7 is: 3
```



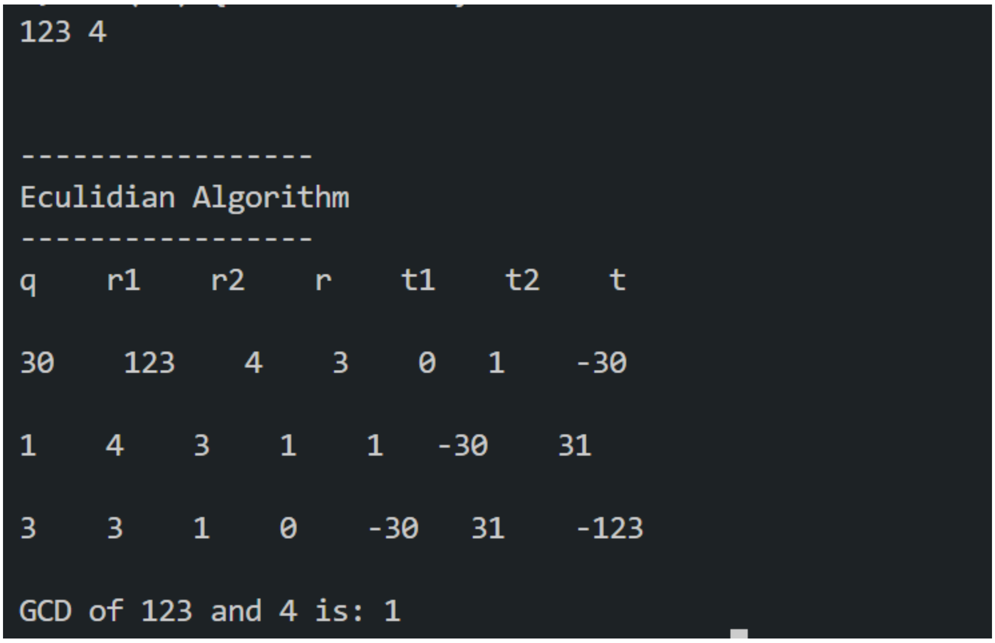
```
13 7


-----------------
Extended Eculidian Algorithm
-----------------

To find Multiplicative inverse first we need to check no's are co-prime or not....

q    r1    r2    r    t1    t2    t

1    13    7    6    0   1    -1

1    7    6    1    1   -1    2

6    6    1    0    -1   2    -13
No's are co-prime hence multiplicative inverse of 13 and 7 is: 2
```



```
$?) { .\euclidian }
1000 128


-----------------
Eculidian Algorithm
-----------------
q    r1     r2     r    t1     t2     t

7    1000     128     104     0    1     -7

1    128     104     24     1    -7     8

4    104     24     8     -7    8     -39

3    24     8     0     8    -39     125

GCD of 1000 and 128 is: 8
```



```
$?) { .\euclidian }
1000 128


-----------------
Eculidian Algorithm
-----------------
q    r1     r2     r    t1     t2     t

7    1000     128     104     0    1     -7

1    128     104     24     1    -7     8

4    104     24     8     -7    8     -39

3    24     8     0     8    -39     125

GCD of 1000 and 128 is: 8
```



```
123 4



-----------------
Eculidian Algorithm
-----------------
q    r1    r2    r    t1    t2    t

30    123    4    3    0   1    -30

1    4    3    1    1   -30    31

3    3    1    0    -30   31    -123

GCD of 123 and 4 is: 1
```