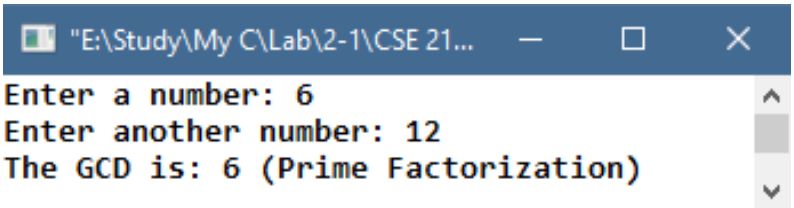
GCD (Prime Factorization)

Code

```
#include<bits/stdc++.h>
using namespace std;

void Intersection(int arr1[], int arr2[], int m, int n)
{
   int mult=1;
   int i = 0, j = 0;
   while (i < m && j < n)
   {
      if (arr1[i] < arr2[j])
         i++;
      else if (arr2[j] < arr1[i])
         j++;
      else
      {
         mult = mult * arr2[j];
         i++;
         j++;
      }
   }
   cout<<"The GCD is: "<<mult<<" (Prime Factorization) "<<endl;
}

int PrimeFactor(int a, int b)
{
   int MAX,len1,len2;
   int PF1[10];
   int PF2[10];
   int divisor1 = 2,divisor2 = 2,i=0,j=0,cPF1=0,cPF2=0;
   while(a>1)
   {
      while(a%divisor1 == 0)
      {
         cPF1++;

         PF1[i++] = divisor1;

         a = a / divisor1;
      }
      divisor1++;
   }
```

```
    while(b>1)
    {
        while(b%divisor2 == 0)
        {
            cPF2++;
            PF2[j++] = divisor2;

            b = b / divisor2;
        }
        divisor2++;
    }

    int m = sizeof(PF1)/sizeof(PF1[0]);
    int n = sizeof(PF2)/sizeof(PF2[0]);

    Intersection(PF1,PF2,m,n);
}

int main()
{
    int a,b,m;
    cout<<"Enter a number: ";
    cin>>a;
    cout<<"Enter another number: ";
    cin>>b;

    PrimeFactor(a,b);

    return 0;
}
```

Output

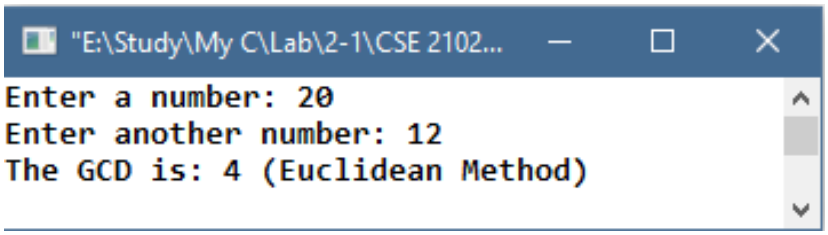


GCD (Euclidean Method)

Code

```
#include<bits/stdc++.h>
using namespace std;

void GCDEuclid(int a, int b)
{
   int r;
   int x = a;
   int y = b;

   while(y !=0)
   {
      r = x % y;
      x = y;
      y = r;
   }
   cout<<"The GCD is: "<<x<<" (Euclidean Method)"<<endl;
}

int main()
{
   int a,b;
   cout<<"Enter a number: ";
   cin>>a;
   cout<<"Enter another number: ";
   cin>>b;

   GCDEuclid(a,b);

   return 0;
}
```

Output



LCM (Prime Factorization)

Code

```
#include<bits/stdc++.h>
using namespace std;

void Union(int arr1[], int arr2[], int m, int n)
{
   int i = 0, j = 0,mult=1;
   while (i < m && j < n)
   {
      if (arr1[i] < arr2[j])
      mult = mult * arr1[i++];

      else if (arr2[j] < arr1[i])
      mult = mult * arr2[j++];

      else
      {
      mult = mult * arr2[j++];
      i++;
      }
   }

   while(i < m)
   {
      mult = mult * arr1[i++];

   }
   while(j < n)
   {
      mult = mult * arr2[j++];
   }
   cout<<"The LCM is: "<<mult<<" (Prime Factorization) "<<endl;
}

int PrimeFactor(int a, int b)
{
   int len1,len2;
   int PF1[3];
   int PF2[2];
   int divisor1 = 2,divisor2 = 2,i=0,j=0,cPF1=0,cPF2=0;
   while(a>1)
   {
      while(a%divisor1 == 0)
      {
```

```
            cPF1++;

            PF1[i++] = divisor1;

            a = a / divisor1;
        }
        divisor1++;
    }


    while(b>1)
    {
        while(b%divisor2 == 0)
        {
            cPF2++;
            PF2[j++] = divisor2;

            b = b / divisor2;
        }
        divisor2++;
    }
    int m = sizeof(PF1)/sizeof(PF1[0]);
    int n = sizeof(PF2)/sizeof(PF2[0]);
    Union(PF1,PF2,m,n);
}

int main()
{
    int a,b,m;
    cout<<"Enter a number: ";
    cin>>a;
    cout<<"Enter another number: ";
    cin>>b;
    PrimeFactor(a,b);

    return 0;
}
```

Output