WORKSHEET – 1

Name: Ruchika Raj

Section/Group: 615 / B

UID: 20BCS9285

Subject: Design and Analysis of Algorithm Lab

Date of Submission: 02.09.2022

Branch: BE CSE

1. Aim/Overview of the practical:

Code and analyse to compute the greatest common divisor (GCD) of two numbers.

2. Task to be done/ Which logistics used:

Task: To find the GCD of two numbers.

Logic used: The GCD of two or more integers is the largest integer that divides each of the integers such that their remainder is 0 or we say that the GCD or HCF of two numbers is the largest number that divides both of them.

GCD of 20,30 = 10 (10 is the largest number which divides 20 and 30 with the remainder as 0)

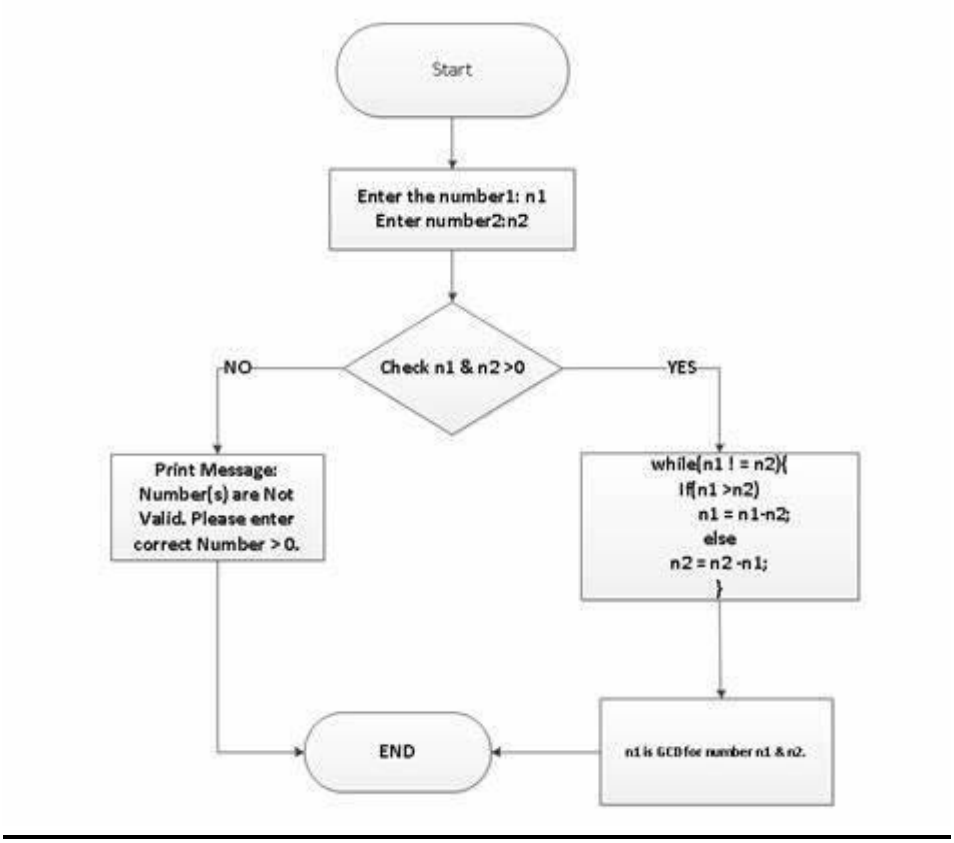
GCD of 42,120,285 = 3 (3 is the largest number which divides 42,120 and 285 with remainder as 0)

Platform Used : Online compiler

3. Algorithm/Flowchart (For programming based labs):

1. **S**tart the program.
2. Take two variables i.e. a and b.
3. If both a and b are 0, gcd is zero i.e. gcd(0, 0) = 0.
4. If one of them is 0, then also gcd will be 0 i.e. gcd(a, 0) = a and gcd(0, b) = b because everything divides 0.
5. If a and b are both even, gcd(a, b) = 2*gcd(a/2, b/2) because 2 is a common divisor. Multiplication with 2 can be done with bitwise shift operator.
6. If a is even and b is odd, gcd(a, b) = gcd(a/2, b). Similarly, if a is odd and b is even, then gcd(a, b) = gcd(a, b/2). It is because 2 is not a common divisor.
7. If both a and b are odd, then gcd(a, b) = gcd(|a-b|/2, b). Note that difference of two odd numbers is even.
8. Repeat steps 3–5 until a = b, or until a = 0. In either case, the GCD is power(2, k) * b, where power(2, k) is 2 raise to the power of k and k is the number of common factors of 2 found in step 3.
9. Print the output.
10. End the program.

Flowchart :



Pseudo Code Of The Algorithm –

Step 1- Let A and B be the two numbers
Step 2- A%B = R
Step 3- let A=B and B=R
Step4 – Repeat step 2 AND 3 UNTIL A%B > 0
Step 5- GCD = B
Step 6- Finish

4. Steps for experiment/practical/Code:

CODE :

```
#include <bits/stdc++.h>
using namespace std;
int gcd(int a, int b)
{
        if (a == b){
           return a;
        }
        if (a == 0){
           return b;
        }
        if (b == 0){
           return a;
        }
        if (~a & 1)
        {
                if (b & 1) {
                   return gcd(a >> 1,b);
                }
                else {
                        return gcd(a >> 1, b >> 1) << 1;
                }
        }
        if (~b & 1) {
           return gcd(a, b >>1);             }
        if (a > b){
         return gcd((a- b) >> 1, b);
        }
   return gcd((b - a) >> 1, a);
}
int main()
{
        int a,b;
        cout<<"ENTER THE FIRST NUMBER :";
```

```
                cin>>a;
        cout<<"ENTER THE SECOND NUMBER :";
        cin>>b;
        cout<<"GCD OF GIVEN NUMBERS IS : "<< gcd(a, b);
        return 0;
}
```

main.cpp

```
#include <bits/stdc++.h>
using namespace std;
int gcd(int a, int b)
{
    if (a == b){
        return a;
    }
    if (a == 0){
        return b;
    }
    if (b == 0){
        return a;
    }
    if (~a & 1)
    {
        if (b & 1) {
            return gcd(a >> 1,b);
        }
        else {
            return gcd(a >> 1, b >> 1) << 1;
        }
    }
    if (~b & 1) {
        return gcd(a, b >>1);       }
    if (a > b){
            return gcd((a- b) >> 1, b);
        }
    return gcd((b - a) >> 1, a);
}
```

```
29  }
30  int main()
31  {
32      int a,b;
33      cout<<"ENTER THE FIRST NUMBER :";
34      cin>>a;
35          cout<<"ENTER THE SECOND NUMBER :";
36      cin>>b;
37      cout<<"GCD OF GIVEN NUMBERS IS : "<< gcd(a, b);
38      return 0;
39  }
40
```

5. Observations/Discussions/ Complexity Analysis:

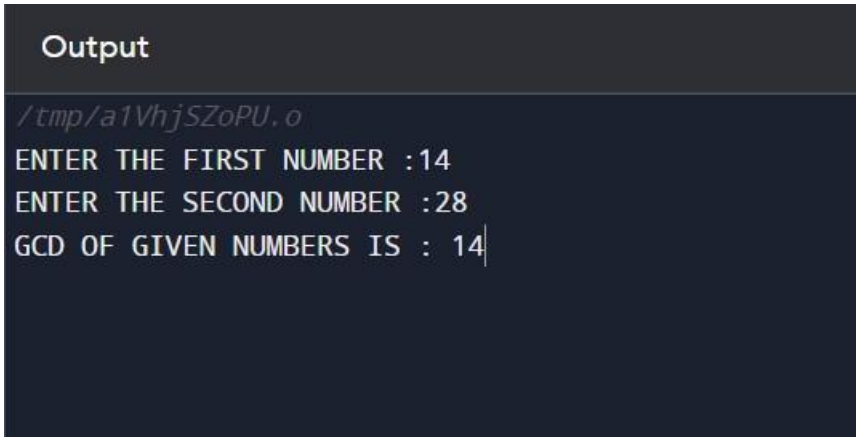
Stein's algorithm or binary GCD algorithm is an algorithm that computes the greatest common divisor of two non-negative integers. This algorithm replaces division with arithmetic shifts, comparisons, and subtraction.

Stein's algorithm is optimized version of Euclid's GCD Algorithm. It is more efficient by using the bitwise shift operator.

Time Complexity : O(N*N) where N is the number of bits in the larger number.

Auxiliary Space : O(N*N) where N is the number of bits in the larger number**.**

6. Result/Output/Writing Summary:



7. Learning Outcomes:

- Learnt about how to calculate gcd of two numbers.
- Learnt about Euclidean Algorithm.
- Learnt how to analyse time and space complexity