NUMBER THEORY

LESSON – 2

Topics to be covered:

1. LCM ,GCD of Array
2. 4 questions
3. Extended Euclidean Theorem (revision) -
4. Prime Numbers (1-n) (1-root (n) -
5. Prime Number (Sieve of Eratosthenes) -

1. LCM OF AN ARRAY

Example : -

Input : {1, 2, 8, 3}

Output : 24

ALGORITHM :-

1. Initialize ans = arr[0].
2. Iterate over all the elements of the array i.e. from i = 1 to i = n-1

At the ith iteration ans = LCM(arr[0], arr[1], ..., arr[i-1]). This can be done easily as LCM(arr[0], arr[1], ..., arr[i]) = LCM(ans, arr[i]). Thus at i'th iteration we just have to do ans = LCM(ans, arr[i]) = ans x arr[i] / gcd(ans, arr[i])

C++ Code:

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

 using namespace std;

```

```c
typedef long long int ll;

int gcd(int a, int b) {
  if (b == 0)
    return a;
  return gcd(b, a % b);
}
ll findlcm(int arr[], int n) {
  ll ans = arr[0];
  for (int i = 1; i < n; i++)
    ans = (((arr[i] * ans)) /
      (gcd(arr[i], ans)));
  return ans;
}

int main() {
  int arr[] = {
    2,
    7,
    3,
    9,
    4
  };
  int n = sizeof(arr) / sizeof(arr[0]);
  printf("%lld", findlcm(arr, n));
  return 0;
}
```

2. GCD of An Array

The GCD of three or more numbers equals the product of the prime factors common to all the numbers, but it can also be calculated by repeatedly taking the GCDs of pairs of numbers.

gcd(a, b, c) = gcd(a, gcd(b, c)) = gcd(gcd(a, b), c) = gcd(gcd(a, c), b)

<u>ALGORITHM</u> :-

```
result = arr[0]
For i = 1 to n-1
  result = GCD(result, arr[i])
```

<u>C++ CODE</u> :-

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

int findGCD(int arr[], int n)
```

```
{
    int result = arr[0];
    for (int i = 1; i < n; i++)
    {
        result = gcd(arr[i], result);

        if(result == 1)
        {
           return 1;
        }
    }
    return result;
}
```

Time Complexity: *O(N * log(M)), where M is the smallest element of the array*

Question Practice

Q1:Find M such that GCD of M and given number N is maximum(M<N)
Given an integer N greater than 2, the task is to find an element M such that GCD(N, M) is maximum.

Examples:

Input: N = 10	**Output: 5**
Input: N = 21	**Output: 7**

Code:

```
int findMaximumGcd(int n)
{
  int max_gcd = 1;
  for (int i = 1; i * i <= n; i++) {
     if (n % i == 0) {
        if (i > max_gcd)
           max_gcd = i;

        if ((n / i != i)  && (n / i != n)   && ((n / i) > max_gcd))
           max_gcd = n / i;
     }
  }
 return max_gcd;
}
```

Q2:Minimum number to be added to minimize LCM of two given numbers

Given two numbers A and B, the task is to find the minimum number that needs to be added to A and B such that their LCM is minimized.

Examples:

Input: A = 6, B = 10 **Output: 2**

Input: A = 5, B = 10 **Output: 0**

Code:

```
vector<int> getDivisors(int diff)
{
  vector<int> divisor;
  for (int i = 1; i * i <= diff; i++) {
           if (i == diff / i) {
        divisor.push_back(i);
        continue;
```

```cpp
    }
        if (diff % i == 0) {
      divisor.push_back(i);
      divisor.push_back(diff / i);
    }
  }
  return divisor;
}

int findTheSmallestX(int a, int b)
{
  int diff = b - a;
  vector<int> divisor= getDivisors(diff);
  int lcm = (a * b) / gcd(a, b);
  int ans = 0;
  for (int i = 0; i < divisor.size(); i++) {
    int x = (divisor[i] - (a % divisor[i]));
    if (!x)
      continue;
    int product = (b + x) * (a + x);
    int tempGCD = gcd(a + x, b + x);
    int tempLCM = product / tempGCD;
    if (lcm > tempLCM) {
      ans = x;
      lcm = tempLCM;
    }
  }
  cout << ans;
}

```

3. Extended Euclidean theorem:

```
          ax + by = gcd(a, b)
Input: a = 30, b = 20
Output: gcd = 10
        x = 1, y = -1
 30*1 + 20*(-1) = 10

Input: a = 35, b = 15
Output: gcd = 5
        x = 1, y = -2
35*1 + 15*(-2) = 5
```

===> $x = y_1 - \lfloor b/a \rfloor * x_1$

$y = x_1$

```
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;
}
```

4. Prime Number

Definition: A number that is divisible by 1 and itself only.

Method 1:

Approach:-

- For any number x,
- From 2 to n-1 check
- If any number can divide it, X is not Prime

Code:-

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

// function check whether a number
// is prime or not
bool isPrime(int n)
{
  // Corner case
  if (n <= 1)
      return false;

  // Check from 2 to n-1
  for (int i = 2; i < n; i++)
      if (n % i == 0)
          return false;

  return true;}
```

```

// Driver Code
int main()
{          int x;
           cin>>x;
   isPrime(x) ? cout << " true\n" : cout << " false\n";
   return 0;
}
```

Method 2:

(Modified and better)

Modification:

Check for roots only upto square root of number. If not root is found. Then Number is Prime.

Code:-

```
bool isPrime(int n)
{
   // Corner case
   if (n <= 1)
       return false;

   // Check from 2 to sqrt n
   for (int i = 2; i *i<= n; i++)
       if (n % i == 0)
           return false;

   return true;}
```

5. Sieve Of Eratosthenes

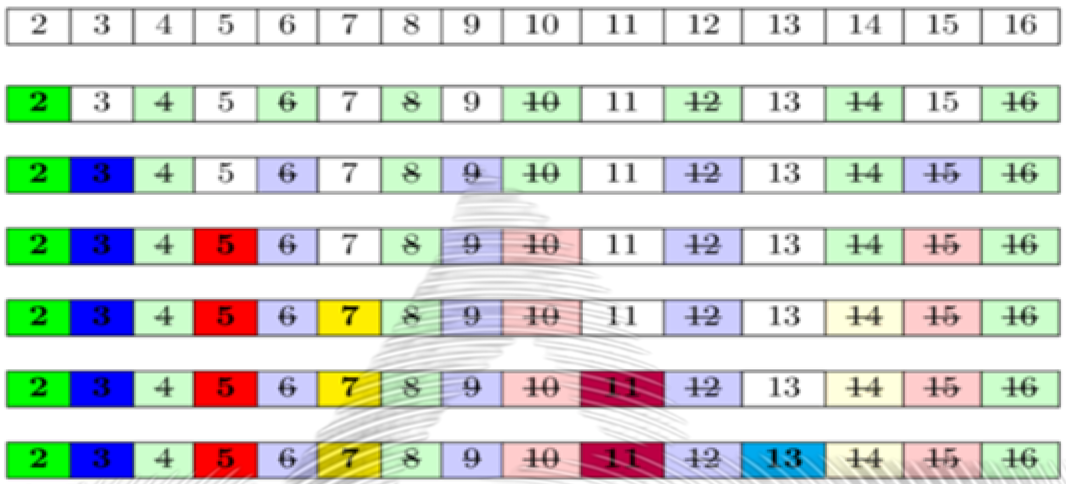
The sieve of Eratosthenes is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million 10^7.

ALGORITHM

The following are the steps used to find prime numbers equal or less than a given integer η.

- List all consecutive numbers from 2 to η, i.e. (2, 3, 4, 5, ..., η).
- Assign the first prime number letter p.
- Beginning with p^2, perform an incremental of p and mark the integers equal or greater than p^2 in the algorithm. These integers will be p(p + 1), p(p + 2), p(p + 3), p(p + 4) ...
- The first unmarked number greater than p is identified from the list. If the number does not exist in the list, the procedure is halted. p is equated to the number and step 3 is repeated.
- The Sieve of Eratosthenes is stopped when the square of the number being tested exceeds the last number on the list.
- All numbers in the list left unmarked when the algorithm ends are referred to as prime numbers.

EXAMPLE N=16



C++ CODE:

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

void SieveOfEratosthenes(int n)
{
    bool prime[n + 1];
    memset(prime, true, sizeof(prime));

    for (int p = 2; p * p <= n; p++)
    {
        if (prime[p] == true)
        {
            for (int i = p * p; i <= n; i += p)
                prime[i] = false;
        }
    }

    // Print all prime numbers
    for (int p = 2; p <= n; p++)
        if (prime[p])
            cout << p << " ";
}
int main()
{
    int n = 30;
    cout << "Following are the prime numbers smaller "
         << " than or equal to " << n << endl;
    SieveOfEratosthenes(n);
    return 0;
}
```

Q3:Find the pair (a, b) with minimum LCM such that their sum is equal to N
Given a number N, the task is to find two numbers a and b such that a + b = N and LCM(a, b) is minimum.

Examples:

Input: N = 15 Output: a = 5, b = 10

Input: N = 4 Output: a = 2, b = 2

Approach: The idea is to use the concept of GCD and LCM. Below are the steps:

1. If N is a Prime Number then the answer is 1 and N – 1 because in any other cases either a + b > N or LCM(a, b) is > N – 1. This is because if N is prime then it implies that N is odd. So a and b, any one of them must be odd and other even. Therefore, LCM(a, b) must be greater than N (if not 1 and N – 1) as 2 will always be a factor.
2. If N is not a prime number then choose a, b such that their GCD is maximum, because of the formula LCM(a, b) = a*b / GCD (a, b). So, in order to minimize LCM(a, b) we must maximize GCD(a, b).
3. If x is a divisor of N, then by simple mathematics a and b can be represented as N / x and (N / x)*(x – 1) respectively. Now as a = N / x and b = (N / x) * (x – 1), so their GCD comes out as N / x. To maximize this GCD, take the smallest possible x or smallest possible divisor of N.

Code:

```
void minDivisior(int n)
{
  if (prime(n)) {
     cout << 1 << " " << n - 1;
  }
  else {
     for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
           cout << n / i << " "<< n / i * (i - 1);
           break;
        }
     }
  }
}

```

Q4:Minimize sum of K positive integers with given LCM

Given two positive integers K and X, the task is to find the minimum possible sum of K positive integers (repetitions allowed) having LCM X.

Input: K = 2, X = 6	Output: 5
Input: K = 3 X = 11	Output: 13

The idea is to represent X in the form of a product of prime powers. Select K prime powers from all the prime powers of X in all possible ways and calculate their respective sums. Finally, print the minimum possible sum among all of them. Follow the steps below to solve the problem:

1. Initialize an array, say primePow[], to store all the prime powers of X.
2. If the length of the array primePow[] is less than or equal to K, then include all the array elements of primePow[] in K positive integers and the remaining element Of K positive integers must be 1.
3. Finally, print the sum of K positive integers