Prime Numbers Numbers having only 2 factors
↳ 1 and number itself.

Eg: 5, 11, 13

count of factors
- == 2 : is prime
- != 2 : not prime.

```
boolean checkPrime ( N) {
    int count = 0
    for( i = 1 ; i <= N ; i++) {
        if ( N % i == 0) {
            count ++
        }
    }
    if (count == 2) { return true }
    else { return false }
}
```

$\boxed{\begin{array}{l} T.C \rightarrow O(N) \\ S.C \rightarrow O(1) \end{array}}$

N = 24.

i	N/i
1	24
2	12
3	8
4	6

$\left\{ \begin{array}{l} i <= N/i \\ i * i <= N \\ i^2 <= N \\ i <= \sqrt{N} \end{array} \right\}$

```
count = 0
for ( i = 1 ; i <= √n ; i++) {
    if (N % i == 0) {
        if (i == N/i) { count += 1 }
        else { count += 2 }
    }
}
if (count == 2) { return true }
else { return false }
```

$\boxed{T.C \rightarrow O(\sqrt{n}) \ , \ S.C \rightarrow O(1)}$

Q1 Given N, get all primes from 1 to N.

N : 10 o/p → [2 3 5 7]

N : 20 o/p → [2 3 5 7 11 13 17 19]

idea-1 Consider all the numbers from 1 to N and check whether they are prime or not.

```
void printallPrimesTillN (N) {
    for ( i = 1 ; i <= N ; i++) {
        if (checkPrime (i) == true){
            print (i)
        }
    }
}
```

T.C → $O(N\sqrt{N})$

S.C → $O(1)$

// Given N = 50, Print all prime no's from 1 to 50.

1 F	2 T	3 T	4 F	5 T	6 F	7 T	8 F	9 F	10 F
11 T	12 F	13 T	14 F	15 F	16 F	17 T	18 F	19 T	20 F
21 F	22 F	23 T	24 F	25 F	26 F	27 F	28 F	29 T	30 F
31 T	32 F	33 F	34 F	35 F	36 F	37 T	38 F	39 F	40 F
41 T	42 F	43 T	44 F	45 F	46 F	47 T	48 F	49 F	50 F

boolean [N+1]

⇓

{0 1 2 ... N} ⇒ N+1.

i = 5 → multiples that we have to consider

2*5, 15, 20, 25, 30

Sieve Of Eratosthenes.

pseudo-code.

```
void getAllPrimesTillN ( N) {
    boolean [] arr = new boolean[N+1];
        arr[ ] → true ;
        arr[0] = false , arr[1] = false ;

    for( i = 2 ; i <= N ; i++ ) {
            if (arr[i] == true) { // i is prime
                //mark all the multiples of i → false.

                for( j = 2*i ; j <= N ; j = j + i) {
                        arr[j] = false
                }
            }
    }

    for( i = 2 ; i <= N ; i++) {
            if ( arr[i] == true) {
                print (i)
            }
    }
}
```

T.C → $O(N \log(\log N))$

S.C → $O(N)$

T.C $\rightarrow$ $\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \frac{N}{11} + \frac{N}{13} + \cdots \frac{N}{p}$

$\{$largest prime $\leq N\}$

$$= N \underbrace{\left\{ \frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots \frac{1}{p} \right\}}_{\text{Sum of reciprocals of prime numbers.}} \Rightarrow N \cdot \log(\log N)$$

idea $\left\{ \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \cdots \frac{1}{N} \right\} \rightarrow \log N$

$$= \int_2^N \frac{1}{x} \cdot dx \longrightarrow \log N$$

Analyse.

$N = 2^{64} \approx 10^{18}$.

$$\log_2 (\log_2 2^{64}) = \log_2 64 = 6.$$

```
void getAllPrimesTillN ( N) {
    boolean [] arr = new boolean[N+1];
        arr[ ] → true ;
        arr[0] = false ,  arr[1] = false ;

    for( i = 2 ; i <= √N ; i++ ) { ⟶ i : [2, √N]
            if (arr[i] == true) { //i is prime
                //mark all the multiples of i → false.

                for( j = i * i ; j <= N ; j = j + i) {
                    arr[j] = false
                }
            }
    }
    //iterate & print
}
```

$\Downarrow$

what if $i = \sqrt{N}+1$

$j = i * i$

$= (\sqrt{N}+1) * (\sqrt{N}+1)$

$= N + 2\sqrt{N} + 1$

$i : \sqrt{N}+1$: no iteration

$i : \sqrt{N}+2$: no "

$i : \sqrt{N}+3$: " "

T.C → $O(\sqrt{N} \log(\log N))$ ×

↓

T.C → $O(N \log(\log N))$

[# todo.
create table.]

We will discuss this in next class's doubt session.

observation

//2 → 4, 6, 8, 10, 12, 14, 16

//3 → 9 {3*2}

//5 → 25 {$\underbrace{5*2}_{2}$, $\underbrace{5*3}_{3}$, $\underbrace{5*4}_{2}$, 5*5}

//7 → 49 {$\underbrace{7*2}_{2}$, $\underbrace{7*3}_{3}$, $\underbrace{7*4}_{2}$, $\underbrace{7*5}_{5}$, $\underbrace{7*6}_{2}$}

//i → (i*i)

2nd part.

Given N. Print the smallest prime factor for all the no's from 2 to N.

N = 10.

2	3	4	5	6	7	8	9	10
↓	↓	↓	↓	↓	↓	↓	↓	↓
2	3	2	5	2	7	2	3	2.

N = 50

1	2	3	4	5	6	7	8	9	10
1	2	3	2	5	2	7	2	3	2
11	12	13	14	15	16	17	18	19	20
11	2	13	2	3	2	17	2	19	2
21	22	23	24	25	26	27	28	29	30
3	2	23	2	5	2	3	2	29	2
31	32	33	34	35	36	37	38	39	40
31	2	3	2	5	2	37	2	3	2
41	42	43	44	45	46	47	48	49	50
41	2	43	2	3	2	47	2	7	2

spf [N+1]

pseudo-code.

```
int spf[N+1]        // initally, spf[i] = i

for( i=2 ; i <= √N ; i++){
        if ( spf[i] == i ) {
            for( j = i*i ; j <= N ; j = j+i ){
                if ( spf[j] == j){
                    spf[j] = i
                }
            }
        }
}
```

T.C → $O(N \log(\log N))$

S.C → $O(N)$

Prime Factorization

[representing a number in multiples of powers of prime numbers.]

2	48
2	24
2	12
2	6
3	3
	1

$48 = 2 * 2 * 2 * 2 * 3$

$48 = 2^4 * 3^1$

no. of factors $= (4+1) * (1+1)$
$= 5 * 2 = 10$

no. of factors are
$\Rightarrow 1, 2, 3, 4, 6, 8, 12, 16, 24, 48$

2	300
2	150
3	75
5	25
5	5
	1

$300 = 2^2 * 3^1 * 5^2$

Q1 Given N, assume prime factorisation. Calculate no. of divisors/factors.

$N = P_1^{a_1} * P_2^{a_2} * P_3^{a_3} \cdots P_x^{a_x}$

① no. of divisors =
$(a_1+1)(a_2+1)(a_3+1)\cdots(a_x+1)$

② Sum of all divisors →

③ Product of all divisors

Google the result.

Q:) Given N. for all the numbers from 1 to N. Get no. of factors for all numbers from [1-N]

N = 10

1	2	3	4	5	6	7	8	9	10
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	1	1	1	1	1	1	1	1	1
	2	3	2	5	2	7	2	3	2
			4		3		4	9	5
					6		8		10

no. of factors = { 1 2 2 3 2 4 2 4 3 4 }

N = 500

↳ for all no's from 1 → 500, consider prime factorisation

↳ spf[501] //Create this array.

// Say x = 360.

2	360
2	180
2	90
3	45
3	15
5	5
	1

, list<int> l

l: | 2 | 2 | 2 | 3 | 3 | 5 |

⇓

$2^3 * 3^2 * 5^1$

$\Rightarrow (3+1) * (2+1) * (1+1)$

$= 4 * 3 * 2$

$= 24$

pseudo-code -

Create & fill spf[N+1]; → $N \log(\log N)$

```
for ( i = 1 ; i <= N ; i++) {
        list <int> p ;
        x = i
        while ( x > 1 ) {
            x = x / spf[x]
            p.add ( spf[x]);
        }
```

→ storing prime-factorization of every number.

→ $\log_2 N$

N

// in formula → we are dealing with power of prime factors.

l = 2 2 2 2 3 3 3 5 5

ans = 1 , c = 0 1 2 3 4 1 2 3

```
        ans = ans * (c+1)

        print(ans);
}
```

todo

$\log_2 N$

T.C → $N \log(\log N) + N \log N$

T.C →	$O(N \log N)$
S.C →	$O(N)$

gcd of entire array.

```
ans = arr[0]
for( i = 1 ; i < N ; i++){
        ans = gcd( ans , arr[i])
}
return ans;
```

ans = arr[i] → 1 iter^n

gcd(1, arr[i] → 1 iter^n.

arr[] → { 64 64 40 16 24 14 19 21 25 }

ans → 64 → 64 → 64 → 8 → 8 → 8 → 2 → 1 → 1

gcd(64, 40)
= gcd(40, 24)
= gcd(24, 16)
= gcd(16, 8)
= gcd(8, 0)

gcd(14, 8)
↳ gcd(8, 6)
↳ gcd(6, 2)
↳ gcd (2, 0)

gcd(19, 2)
↳ gcd(2, 1)
↳ gcd(1, 0)

+ + + + = N

arr[0] → 1

+ + = $\log_2 arr[0]$

{ T.C = $N + \log_2 arr[0]$ }

$\boxed{T.C \to O(N)}$

for large values of N.