Prime numbers

A number which is divisible by itself and 1.

Eg:- 2, 3, 5 . . . . . .

General logic to find prime:-

```
for(i=2; i<N; i++){
    if(N%i){
        Not prime
    }
    else
        prime
```

NOTE:- This loop can only be done till $i < \sqrt{N}$

```
for(i=2; i<sqrt(N); i++)
{
}
```

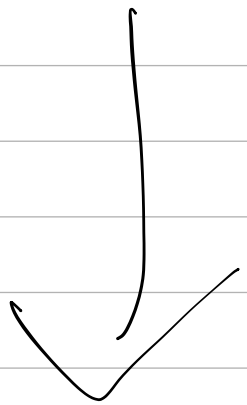
USING WHILE LOOP :-

```
static boolean isPrime(int n){
    if(n<=1){
        return false;
    }
    int c = 2;
    while(c*c <= n){
        if(n % c == 0){
            return false
        }
        c++;
    }
}
```

Q). Find all the numbers less than or equal to N, that are prime

Soln:-

```
psvm(String[] args){
    int n = 40;
    boolean ans = false;
for(int i=2; i<=n; i++){
    ans = isPrime(i);
if(ans == true)
{
    sout(i);
}
}
```



```
static boolean isPrime(int n){
    if (n<2){
        return false;
    }
    else if (n>=2){
        for(int i=2; i<=Math.sqrt(n)
                                i++)
```

{	if (n%i==0)	
2	True	return false;
3}	True	
}		
else		
	return true;	

```
}
```

OUTPUT:

2
3
5
7
:
:
37.

Sieve of Eratosthenes

Here we we use a Boolean array to store true if it is not prime ,false if that index is prime

CODE:-

```
static void sieve(int n, boolean[] primes){

for(int i = 2; i*i <= n; i++){

    if(!primes[i]){
    for(int j= i*2; j<=n; j+=i)
{
        primes[j] = true;
    }
}
}

    for(int i = 2; i<=n; i++){
        if(!primes[i]){
            sout(i);

}
}
```

For example :- n = 10.

2 3 4 5 6

7 8 9 10

0 -> true

X -> false.

i.e, primes [2] = false

primes [3] = false

primes [4] = true.

:

Space complexity = O(n).

n -> size of boolean array.

Time complexity :-

NOTE :- The numbers from 0 to n, which are divisible by a, is $\frac{n}{a}$ numbers

Eg :- from 0 to 40, the total numbers divisible by 2 are $\frac{40}{2}$ = 20 numbers.

Divisible by 3, = $\frac{40}{3}$ numbers.

∴ In above program,

for j loop runs,

In first iteration,

$\frac{n}{2}$ times,

Second iteration,

$\frac{n}{3}$ times) $\frac{n}{5}$ times soon,

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \cdots$$

$$n\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots\right).$$

Taylor series

Harmonic progression for primes.

Formula : $\log(\log N)$.

Total time complexity :

$O(N * \log(\log N))$

Finding square of a number

Binary search technique. Eg:- N = 36.

0 (18) 36.

↓ middle.

Logic :- Here, 18 x 18 > 36, so, end = mid - 1
else start = mid + 1.

↓ This is for perfect square numbers.

Suppose $\sqrt{40}$:- First find integer value of square root i.e 6.

Then. check $6 \times 6 < 40$

$+0.1$

$6.1 \times 6.1 < 40$

$+0.1$

$6.2 \times 6.2 < 40$

$+0.1$

$6.3 * 6.3 < 40$

$+0.1$

$6.4 \times 6.4 > 40 \rightarrow$ So, 6.3 is the answer

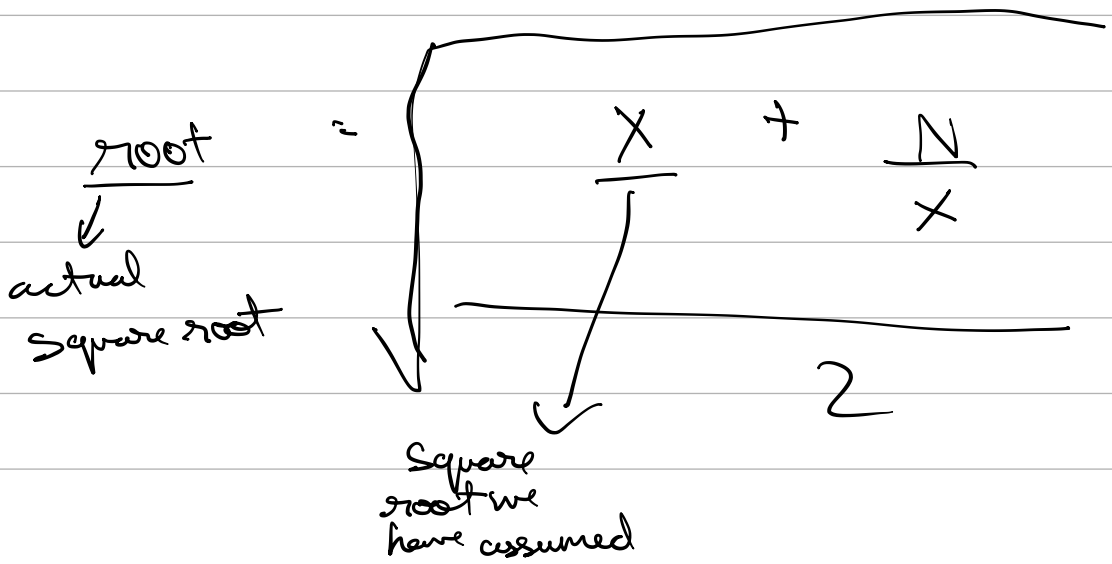
```java
public class BinarySearchSQRT {
    public static void main(String[] args) {
        int n = 40;
        int p = 3;

        System.out.println(sqrt(n, p));
    }

    static double sqrt(int n, int p) {
        int s = 0;
        int e = n;

        double root = 0.0;

        while (s <= e) {
            int m = s + (e - s) / 2;

            if (m * m == n) {
                return m;
            }

            if (m * m > n) {
                e = m - 1;
            } else {
                s = m + 1;
            }
        }
        double incr = 0.1;
        for (int i = 0; i < p; i++) {
            while (root * root <= n) {
                root += incr;
            }

            root -= incr;
            incr /= 10;
        }

        return root;
    }
}
```

Here n -> no. to find square root.

p -> precision (i.e no. of decimal digits required in answer).

Time complexity :- O(logN).

Newton Raphson Method to find square root



Proof

Suppose we assumed 'x' correctly,

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

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

$\sqrt{N} = \sqrt{N}$ Correct.

Time Complexity = $O((\log N) F(N))$

$F(N)$ → cost of calculating $\frac{f(x)}{f'(x)}$ with N digits precision.

JUST TRY TO REMEMBER, THIS IS COMPLEX MATHS

Code to implement Newton method

```java
public class NewtonSQRT {
    public static void main(String[] args) {
        System.out.println(sqrt(n: 40));
    }
    static double sqrt(double n) {
        double x = n;
        double root;
        while (true) {
            root = 0.5 * (x + (n/x));

            if (Math.abs(root - x) < 0.5) {
                break;
            }

            x = root;
        }
        return root;
    }
}
```

This 0.5 is to check the difference b/w actual root and assumed root, if it is less than (0.5.) then break and root is correct. (0.5 basically determines the

precision).

Factors of a number

Generic method:-

```
n=20;
for(i=1;i<=n;i++)
{
    if(n%i==0)
        sout(i);
}
```

Time complexity = O(n)

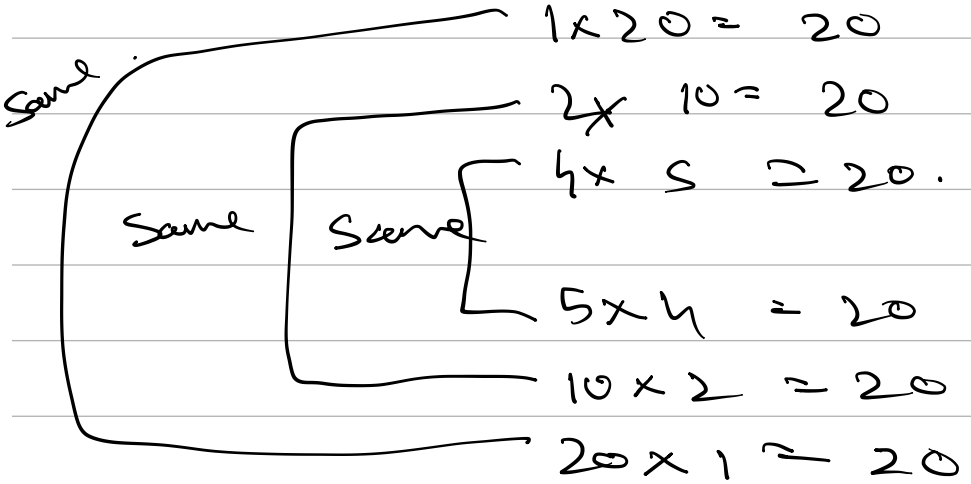
Optimised Method:-

```
for(int i=1;i<=Math.sqrt(n);i++)
{
    if(n%i==0){
        if(n/i==i){
            sout(i);
        }else{
            sout(i+" "+n/i);
        }
    }
}
```

Time complexity = $O(\sqrt{n})$

Logic:- If n = 20.

We know,



Here if $i = 1$, $\frac{n}{i} = 20$

So, take both i and $\frac{n}{i}$ as factors and print.

Output:- 1 20 2 10 4 5

To get output in order:-

```
ArrayList<Integer> list = new ArrayList<>();
for (int i = 1; i <= Math.sqrt(n); i++) {
    if (n % i == 0) {
        if (n/i == i) {
            System.out.print(i + " ");
        }else {
            System.out.print(i + " ");
            list.add(n/i);
        }
    }
}
for (int i = list.size() - 1; i >= 0; i--) {
    System.out.print(list.get(i) + " ");
}
}
```

→ Here we use (list) to store $\left(\frac{n}{i}\right)$ values and print it later after i values, to get the result in order.

Now Output:- 1 2 4 5 10 20

→ In order.

Properties of modulo %

*). $(a+b) \% m = ((a \% m) + (b \% m)) \% m$

*). $(a-b) \% m = ((a \% m) - (b \% m) + m) \% m$

*). $(a*b) \% m = ((a \% m) * (b \% m)) \% m$

*). $\left(\frac{a}{b}\right) \% m = ((a \% m) * (b^{-1} \% m)) \% m$

Here $b^{-1} \% m$ is multiplicative modulo inverse.

Eg:- $(6 \times y) \% 7 = 1$

$y \rightarrow$ multiplicative modulo inverse for 6 & $y = 6$

$\because (6 \times 6) \% 7 = 1$

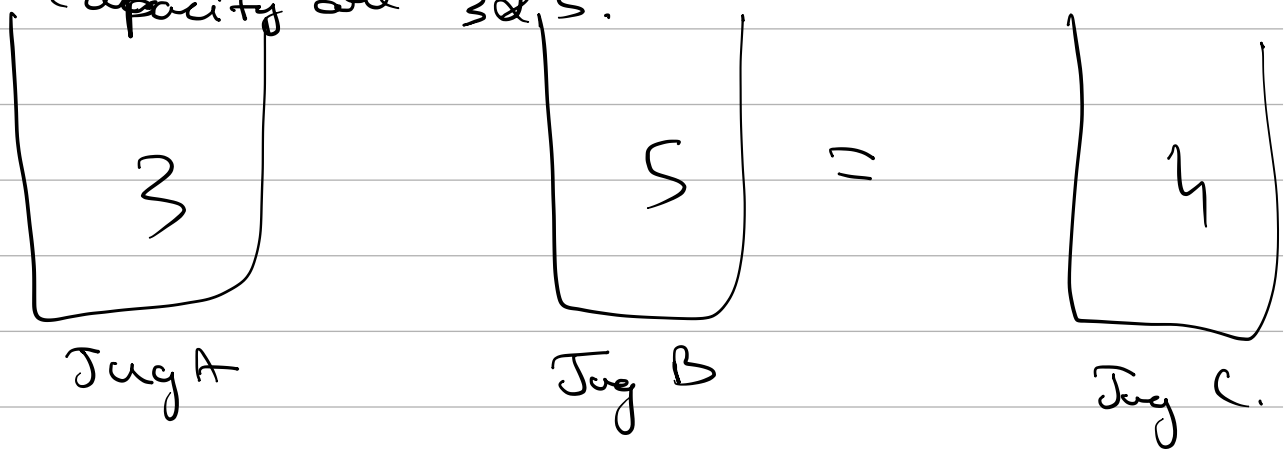
NOTE:- b & m are co-primes i.e No numbers is common except 1

Eg:- Above 6 & 7 are co-primes.

*). $m \,\%\, m = 0 \quad \forall x \in$ +ve integers.

Die hard movie example

How to get 4 litre of water in C if A & B capacity are 3 & 5.



How to get exactly 4 litres of water to JugC using jugs A & B with 3 & 5 ltr capacity.

1st $\Rightarrow$ $\overset{A\ B}{(3,0)} \rightarrow \overset{A\ B}{(0,3)}$

2nd $\Rightarrow$ $\overset{A\ B}{(0,3)} \rightarrow \overset{A\ B}{(3,3)} \rightarrow \overset{A\ B}{(1,5)}$

$\downarrow$

$\overset{A\ B}{(0,1)} \leftarrow (1,0)$ B upty Jug B

3rd $\rightarrow$ $(0,1) \rightarrow (3,1) \rightarrow (0,④)$

A B we got 4.

Here we are continously transfering water from Jug A to B to get 4 in B and that can be transfered to Jug C.

Jug A → s' times (Water being transfered)

Jug B → s^2 times.

remainder (r) = $as' - bs^2$

(r: Volume we get at last; as': volume of A; bs^2: vol of B.)

i.e, $r = as' + (-bs^2)$

Note:- s' & s^2 are not power 1 and 2 it is s_1 and s_2 times

$r = s'a + t'b - t'b - bs^2$

$r = L - (t' + u)b^2$

Here, $L = s'a + t'b$

$s'a = L - t'b.$

If $t'+u \neq 0 \Rightarrow (r<0 \text{ or } r>b)$

which is not true as volume remaining cannot be <0 or $>b$.

$t'+u=0 \Rightarrow u=-t$

$r = s'a + t'b = L$ $\left(\because t'+u=0\right)$

↓

of the form.

$r = ax + by$

Put x & y as integers, what is the minimum value we can get.

Example:- $3x+5y=4$

If $x=-3$, $y=2$.

$3x+5y=1$

↓

the minimum value of $3x+5y$ we can get is 1.

Note:- This (1) is called HCF of 3 & 5.

HCF / GCD :-

HCF of a & b = minimum +ve value of equation $ax+by$ where $x, y \rightarrow$ integers.

Definition:-

Minimum value of $ax+by$ where x, y can be any integer.

$\min(3x+9y)$

Put $x=-2$, $y=1$

$= 3(-2) + 9(1)$

$= -6 + 9$

$= 3 \rightarrow$ HCF.

Practical meaning :-

If there is jug of 3L and 6L can we get 9L of water.

Eg:- $3x+6y=9$.

$3(x+2y)=9$

$x+2y=3$ → integer, so we can get 9L of water.

Eg-2:- If we have 2L and 4L of jug, can we get 5L of water.

$2x+4y=5$

$2(x+2y)=5$.

$x+2y=2.5$ → not an integer, so cannot get 5L of water.

Eg-3:- $3x+5y=17$

$1\ (3x+5y)=17.$

$3x+5y=17$ → Integer

Can form ✓

Euclid Algorithm to find GCD

$$GCD(a,b) = GCD(\text{remainder}(b,a), a)$$

Eg:- $GCD(105, 224) = gcd(rem(224,105), 105)$

$= gcd(14, 105)$

↳ $gcd(rem(105,14), 14)$

↳ $gcd(7, 14)$

↓

Goes on.

Linear representation of GCD of 105, 224 is

$$224x + 105y = \text{HCF}.$$

↓ minimum HCF value.

Logic:- as we go, on doing $b \% a$

i.e remainder (b, a), at some point $b \% a = 0$

$=$

that time we return the other number.

For example:-

GCD $(2, 4)$

$= gcd(rem(4, 2), 2)$

$\hookrightarrow gcd(0, 2)$

Here we return 2, as GCD.

Code in recursion GCD Euclidean method

```java
public class GCD_LCM {
    public static void main(String[] args) {

    }

    static int gcd(int a, int b) {
        if (a == 0) {
            return b;
        }
        return gcd( a: b%a, a);
    }
}
```

→ when b%a i.e a^th position becomes 0, return b.

LCM

→ LCM of 2 numbers is a minimum number divisible by both the numbers.

Formula:-

$$L.C.M = \frac{a \times b}{HCF}$$

HOW DID WE GET THIS FORMULA?

Suppose $d \rightarrow$ is GCD of a & b.

Consider, $f = \frac{a}{d}$ and $g = \frac{b}{d}$.

Then $LCM = f \times g \times d$

$$\Rightarrow \frac{a \times b}{d^2} \times d$$

$$L.C.M \Rightarrow \frac{a \times b}{d}$$

Example :- $a = 9, \quad b = 18.$

$d = 9 \rightarrow$ H.C.F of a & b.

$$f = \frac{a}{d} = \frac{9}{9} = 1$$

$$g = \frac{b}{d} = \frac{18}{9} = 2.$$

We have removed highest factor

in each number a & b, by dividing it by 'd'.

$\therefore$ L.C.M = f * g * d.

$= 1 \times 2 \times 9$

L.C.M = 18

Logic:- L.C.M is a number divisible by both a & b, so, it should have 'a x b'

But we are removing highest common factor, so,

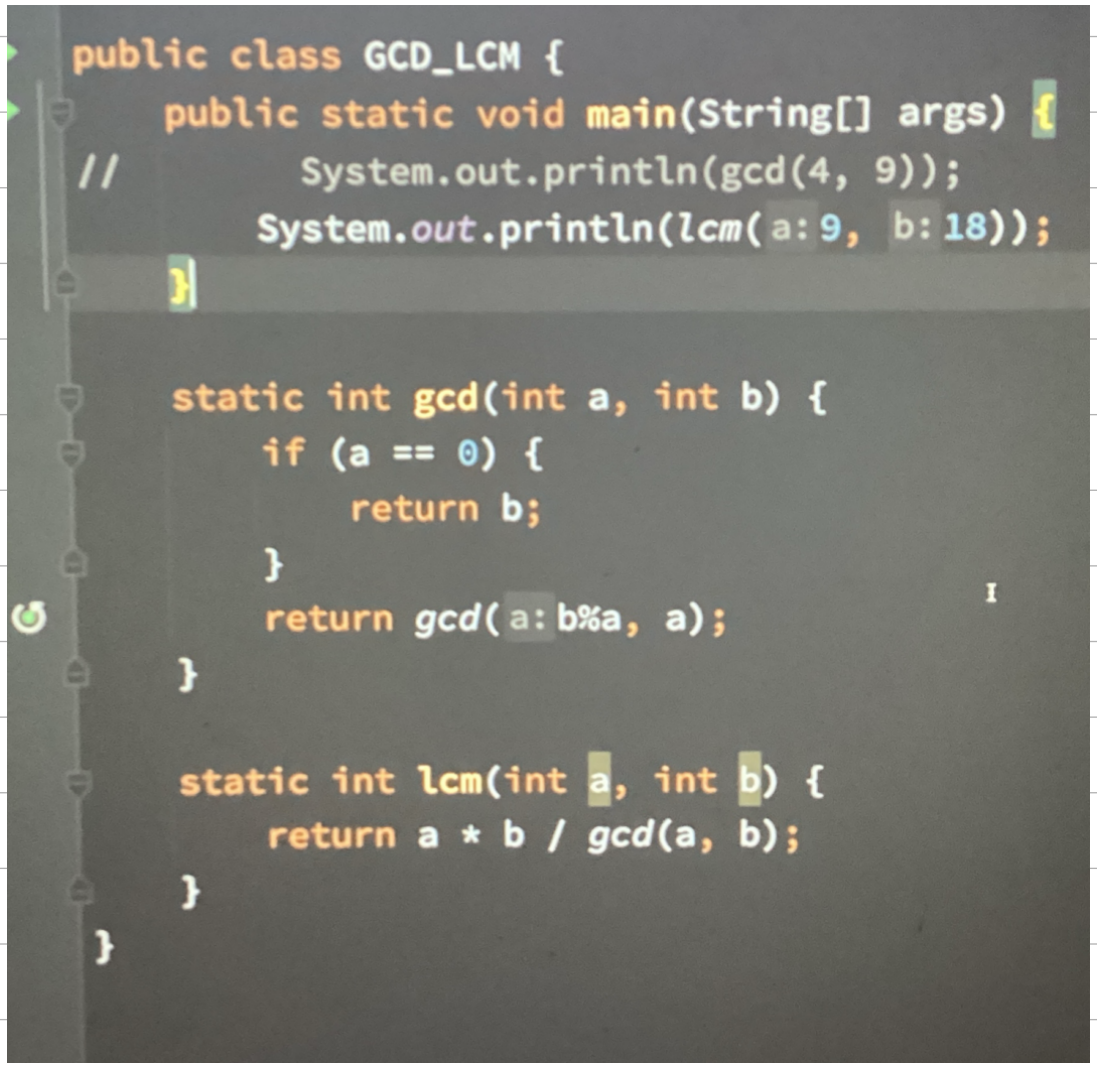
$$L.C.M = \frac{a \times b}{H.C.F}$$

NOTE:-

$$L.C.M = \frac{a \times b}{H.C.F}$$

$$\Rightarrow \boxed{L.C.M \times H.C.F = a \times b.}$$

Code



```java
public class GCD_LCM {
    public static void main(String[] args) {
//        System.out.println(gcd(4, 9));
        System.out.println(lcm( a: 9,  b: 18));
    }

    static int gcd(int a, int b) {
        if (a == 0) {
            return b;
        }
        return gcd( a: b%a, a);
    }

    static int lcm(int a, int b) {
        return a * b / gcd(a, b);
    }
}
```

↓

Output :-

LCM = 18.

We know, $a * b \Rightarrow 9 \times 18 = 162$.

$\frac{a*b}{HCF} \Rightarrow \frac{162}{9} = 18$