Maths Required for DSA.

① Number of digits in a number.

1 2 3 4 5 ⟶ 5 digits

Program →
```
int countDigit(int number){
    count = 0;
    while(number!=0){
        number= number/10;
        count++;
    }
    System.out.println("count" + number);
}
```

② Recursion. ⟶
```
int countDigit(int number){
    if(number==0){
        return
    }
    else{ number = number/10;
    return(1+countDigit(number));
    }
}
```

Log program (bonus) ⟶
```
int countDigit(int n){
    return(floor(log10(n)+1));
}
```

② Arthemetic & Geometric progressions

→ 2, 4, 6, 8, 10 . . . . .

① take $a = 2$

$d = diff = 2$

$a+d,\ a+2d,\ a+3d\ \ldots\ a+(n-1)d.$

② Sum $= \frac{n}{2}(2a+(n-1)d)$.

③ $\left(\frac{n}{2}\right) * (d) =$ Sum of even − Sum of odd.

Geometric

→ 2, 4, 8, 16, 32 . . . .

③

$a = 2$

$r = 2$

$a, ar, ar^2, ar^3 \ldots ar^{n-1}$

② Sum $= a(\cancel{1}-r)$.

③ Quadratic equations

⇒ $ax^2 + bx + c = 0$

$b = \alpha + \beta$

$c = \alpha\beta$

$b^2 - 4ac = 0$ } equal roots.

< 0 } imag

> 0 } distinct real..

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

④ mean & median

mean $= \frac{\text{sum}}{n}$

median = mean of middle two numbers, or middle number.

⑤ Prime num → 1 & itself only divisible.

$6n+1$ } every prime num
$6n-1$ ✓
except 2 or 3

⑥ LCM & HCF

28 36

factors of 28 → 1, 2, 4, ---- 28

Highest comm factor of 28 & 36 → 4.

LCM →

$$
\begin{array}{r|l}
2 & 28, 36 \\ \hline
2 & 14, 18 \\ \hline
3 & 7, 9 \\ \hline
 & 7, 3
\end{array}
$$

$2^2 \times 3^2 \times 7$

$9 \times 4 \times 7$

→ 9×28 → 252.

⑦ Factorial

$5! \rightarrow 5 \times 4 \times 3 \times 2 \times 1$.

⑧ P & C → in aptitude.

⑨ Mod Arthemetic (%)

→ $10^9 + 7$ module → mostly answer eq in this format.

to prevent integer overflow.

21 % 7 → 0

21 % 4 → 1.

4 % 21 → 4

Problems in Maths

1) Palindrome
2) Count digits
3) Reverse digits
4) Find LCM and HCF.
5) Check for Prime num.
6) Find factorial of a number.

Programs:

```
int Reverse (int number){
    int ones;
    int rev;

  while (number != 0){

      ones = number%10;
      rev = (rev*10) + ones;
      number = number/10;
  }

  return rev;

}
```

for. palindrome.
compare at last
return (temp == rev).
where temp = number.

efficient till $\sqrt{n}$ bcz prime in pair.

you can check 1 to n loop where n%i==0 (Naive method)

```
bool Prime (int n){

  if (n==2 || n==3){
      return true;
  }

  if (n%2==0 || n%3==0){
      return true }.

for (int i = 5; i*i <= n; i+=6)
  { if(n%i==0 || n%(i+2)==0) return false;
  }
  return true;
}
```

Program.

```
int factorial(int n){
  int fact = 1
  for (int i=1; i<n; i++)
    { fact = fact * i
    }
  return fact;
```

O(n)
O(1)

Recursion factorial.

```
int fact(int n){
  if (n == 0){return 1;}
  else{
      return (n * fact(n-1))
```

O(n)
O(n)
Extra overhead. }

7) Trailing zeroes in a factorial

① 5 2 10 multiple 10 5

Adv sol → ∴ Count 2 & 5 in using prime fact of number fact.

Note → No: of 5 always less than 2, ∴ Just count 5.

② Naive solution

After computing factorial,
→ count zeroes in result using (fact % 10 == 0) then reset..

O(n) but very overflow.

---

8) Program for ⑦

O(logn)

```
int countTrailingZeroes(int n)
{ int res = 0;
  for (int i = 5; i <= n; i = i*5)
  {
     res = res + n/i;

  }
  return res;
}
```

---

⑧ ④ HCF & LCM

a & b → start with smaller number (i = min(a, b)). keep doing i-- until i divides both a & b, } O(min(a,b))
(HCF Naive)

→ (HCF using Euclid algorithm) →

```
int gcd(int a, int b){
  while (a != b)
  { if (a > b)
      {a = a - b;}
    else { b = b - a }
  }
  return a;
}
```

gcd (a, b) = gcd (a - b, b) where a > b

Additional Problems in maths

LCM & HCF

```
int HCF( int a, int b){
    if (b==0){
        return a;
    }
    return HCF(b, a%b);
}

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

Checking for Prime factors

3 methods already discussed for checking prime (or not.

- n = 12
  Prime facts → 2, 2, 3.

- Ideas used
  1) Divisor appear in pairs
  2) Number can be written as prod of primes.

- Algo

```
Void primefactors (int n)
{ if(n<=1) return;
  for(int i=2; i*i <= n; i++)
  { while(n%i==0){
        print(i);
        n = n/i;
    }
  }
  if(n>1){print(n)}
```

Printing Divisors

```
Void PrintDivisors(int n)
{ for(int i=1; i*i<=n; i++)
  { if(n%i==0)
    { print(i)
      if(i!=n/i)
      { print(n/i);}
    }
  }
}
```

Sieve of Eratosthenes

- Given a number n, find all prime numbers smaller than equal to n.
-

Algo

```
SieveofEra(int n){
  if(n==3)
    {print(2);}

  for(i=3; i<n; i++)
  { if(i%2==0 || i%3==0)
      {continue;}
                bool x;
    else {x=Checkprime(i);}
    if(x=true){print i;}
    }
  }
}
```

$O(n\sqrt{n})$.

Sieve of Era

Marks false of all multiples of current number.

Only prime left as true at last.

$O(n \log\log n)$.

Computing power → in (logn) shown. time, using recursion. but $O(n)$ space

↳ Naive $O(n)$.

→ Iterative soln shown with $O(1)$ space only.