Basic Math concepts

Digit:

% ⟹ remainder

/ ⟹ Quotient

Eg: N = 7789 %10 ⇒ 9

↳ /10 ⇒ 778 %10 ⇒ 8

↳ /10 ⇒ 77 %10 ⇒ 7

↳ /10 ⇒ 7 %10 ⇒ 7

↳ /10 ⇒ 0

Count Digit:

```
#include <bits/stdc++.h>
using namespace std;
int countdigit(int n)
{
  if(n==0)
  int countdigit = 0;
  while(n>0)
  {
    int lastdigit = n%10;
    countdigit
```

Count digit:

```
#include <bits/stdc++.h>
using namespace std;
int count (int n)
{
    int cnt = 0;
    while(n>0)
    {
      int lastdigit = n%10;
      cnt = cnt+1;
      n = n/10;
    }
    return cnt;
}
int main()
{
    int n;
    cin >> n;
    cout << count(n)
    return 0;
}
```

Alternate Method:

```
#include <bits/stdc++.h>
int count (int n)
int cnt = (int) log10(n)
          + 1);
return cnt;
}
```

TC $\Rightarrow O(\log_{10}(N))$

because it (1) divides by 10 so Time complexity is $\log_{10}$

Reverse the number:

Eg: 7789

LD = N % 10

LD = 9

N = N / 10

N = 778

rev = (0 x 10) + 9

rev = 9

LD = N % 10

LD = 778 % 10 $\Rightarrow$ 8

N = N / 10

N = 778 / 10

= 77

rev = (9 x 10) + 8

$\Rightarrow$ 98

```
rev = 0
7789 > 0
while (N>0)
{
  LD = N%10
  N = N/10
  rev = (rev x 10) + LD
```

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
  int n;
  cin >> n;
  int revnum = 0;
  while (n > 0)
  {
    int ld = n % 10;
    revnum = (revnum * 10) + ld;
    n = n / 10;
  }
  cout << revnum;
}
```

Step 1: (63)

n = 7789

while (7789 > 0)

ld = 7789 % 10

ld = 9

r = r * 10 + ld

⇒ 0 * 10 + 9

r = 9

n = n / 10

7789 / 10

n = 778

Step 2:

n = 778

while (778 > 0)

ld = 778 % 10

ld = 8

r = r * 10 + ld

= (9 * 10) + 8

r = 98

Check Palindrom:

```
#include <iostream>
using namespace std;
int main()
{
  int n;
  cin >> n;
  int revnum = 0;
  int revnum;
  int dup = n; // store n in duplicate
```

```
while (n>0)
{
    int ld = n%10;
    revnum = (revnum * 10) + ld;
    n = n/10;
}
if (dup == revnum) cout << "Palindrome";
else cout << "Not";
}
```

(b)

Armstrong Number:

If a number is equal to the sum of its own digit cube.

Eg: $153 \Rightarrow 1^3 + 5^3 + 3^3$

$\Rightarrow 1 + 125 + 27$

$\Rightarrow 153$

```
1
125
27
1
153
```

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int temp = n;
    int p = 0;
```

```
dup = N
sum = 0
while (N>0)
{
lastdigit = N%10
sum = sum + (ld * ld * ld)
N = N/10;
```

```
while (n>0)
{
 int rem = n%10;
 P = P + (rem * rem * rem);
 n = n/10;
}
 If (Temp==P)
 {
  cout << " Armstrong No";
 else
 {
  cout<<" not";
 }
return 0;
}
```

Print all Division:

36 ⇒ 1, 2, 3, 4, 6, 9, 12, 18, 36

36 ⇒ 9

Time Complexity ⇒ O(n)

```
#include <iostream>
using namespace std;
void printdivisor (int n)
{
  for (i=1; i<=n; i++)
    {
```

```
        if (n % i == 0)
        {
            cout << i << " ";
        }
    }
}
int main()
{
    int n;
    cin >> n;
    Printdivisor(n);
    return 0;
}
```

Alternate method: using vector.

```
#include <bits/stdc++.h>
using namespace std;
void Printdivisor(int n)
{
    for (int i = 1; i * i <= n; i++)        // loop
    {
        if (n % i == 0)                     // To find factors
        {
            ls.push_back(i);                // {-, -, ——, }
            if ((n/i) != i)
            {
                ls.push_back(n/i);
            }
        }
    }
}
```

36

n/i

⇒ 36 ⇒ 36

⇒ 36/2 ⇒ 18

⇒ 18/3 ⇒ 6

⇒ 6/6 ⇒ .

```
sort( ls.begin(), ls.end());
for( auto it : ls) cout << it << " ";
}
int main()
{
  int n;
  cin >> n;
  printdivisor(n);
  return 0;
}
```

N = 36

1 x 36
2 x 18
3 x 12
4 x 9
~~6 x 6~~
9 x 4
12 x 3
18 x 2
36 x 1

$\Rightarrow N/i$

$N \times N/i$

$i \times i <= n$

$1 \times 1 <= 36$

$2 \times 2 \Rightarrow 36$

$3 \times 3 \Rightarrow 36$

$4 \times 4 \Rightarrow 36$

$5 \times 5 >=$

$n \% i == 0$ ✗

$\frac{36}{1} \Rightarrow 36$

$\frac{36}{2} \Rightarrow 18$

$\frac{36}{3} \Rightarrow 12$

$\frac{36}{4} \Rightarrow 9$

$\frac{36}{5} = n \% i == 0$ ✗ false

Prime Number Checks:

* A number divisible by 1 and itself.
* It exactly has 2 factors 1 & itself

```
#include <bits/stdc++.h>
using ~~using~~ namespace std;
int main()
{
   int n;
   cin >>n;
   int cnt =0;
   for (int i=1; i*i<=n; i++)
   {
      if (n%i==0)
      {
         cnt++;
         if ((n/i) != i) cnt++;
      }
   }
   if (cnt==2) cout << "Prime";
   else cout << "not";
}
```

3
3 | 3
3
0

n = 3
i = 1 ; 1*1 <= 3
if (3%1 == 0)
{
1 ;
if (3(1) != 1)
2
⇒ cnt == 2
Prime

n = 8
i = 1 ; 1*1 <= 8
if (8%1 == 0)
⇒

GCD (Greatest common Divisor) (or) HCF (Highest common Factor)

⟹ Find the highest common factor between 2 numbers.

Eg: GCD between 36 and 60

```
2 | 36
2 | 18
3 | 9
3 | 3
  | 1
```

36 ⇒ 2 × 2 × 3 × 3

```
2 | 60
2 | 30
3 | 15
5 | 5
  | 1
```

60 ⇒ 2 × 2 × 3 × 5

$2 \times 2 \times 3 = 12$

GCD of 36, 60 = 12

Euclidean Algorithm:

$gcd(n_1, n_2) \Rightarrow gcd(n_1 - n_2, n_2)$

where $n_1 > n_2$