Doubt class

① Power of 2

M-1 :-

1 → 1

2 → 10

4 → 100

8 → 1000

16 → 10000

$\xrightarrow[\text{from each}]{\text{subtracting 1}}$

0 → 0

1 → 1

3 → 11

7 → 111

15 → 1111

Exa: 4 & 3 ⇒

$$\begin{array}{r} 100 \\ 011 \\ \hline 000 \end{array}$$

∴ 4 is a power of 2.

If [n & (n-1)] == 0 ⇒ then that particular no is power of 2

- → Powers of 2 have only one '1' in binary.
- → subtracting 1 flips all the bits after that 1.
- → ~~Addin~~ AND(&)-ing them becomes zero.

$\boxed{\text{T.C : O(1)}}$

Exa: 9 & 8

$$\begin{array}{r} 1000 \\ 1001 \\ \hline 1000 \end{array} \Rightarrow \neq 0$$

∴ 9 is Not a power of 2.

M-2:

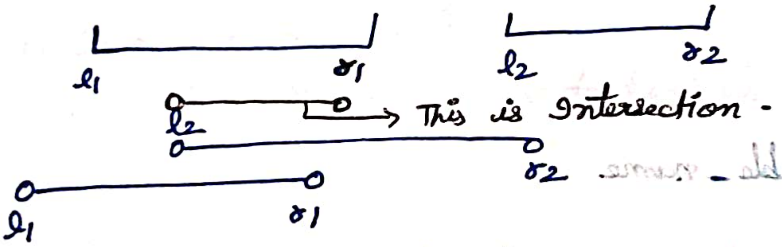
```
while (n%2 = =0)
        n = n/2;
if (n= = 1) return 1;
return 0;
```

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

M-3: if ((n/2) % 2 = = 0) → This is wrong approach.

M-4: If the binary representation of the number has only one '1', then it is a power of 2.

2. Two Intervals :-



$l_2 = \max(l_1, l_2)$ → starting pt

$r_1 = \min(r_1, r_2)$ → Ending pt

```
if (starting <= ending)
        intersection exist
else
        intersection does not exist
```

3. GCD

M-1: Use in-built STL func^n :- gcd (a, b)

M-2: Use Euclidean GCD Algorithm.

4. Lucky Numbers :-
↳ A no. consisting only of 4 & 7

Traverse every digit of the no. using modulo 10 and check if the digits are 4 & 7 only and stop if any other no. found.

Code :-

```
int lucky (int x){
  while (x){
    int digit = x%10;
    if (digit != 4 && digit!= 7) return 0;
    x/= 10;
  }
  return 1;
}
void solve ~~(int a,~~( ){
  int a,b;
  cin >>a >>b;
  for (int i=a;i<=b;i++){
    if (lucky (i)){
      cout << i << " ";
    }
  }
}
```

5. sum of odd no. b/w X & Y :-

If X is even → start = X+1
If X is odd → start = X+2
→ start <= Ending

Y → Even → Ending = Y-1
Y → odd → Ending = Y-2

$$S_n = \frac{n}{2}(a+l) \rightarrow AP$$

Code :-

```
int x,y;
cin >>x>>y;
int starting = min (x,y); ending = max (x,y);
if (x%2) starting += 2;
else starting +=1;
if (y%2) ending -= 2;
else ending -= 1;
```

$\boxed{T.C : O(1)}$

```
if (starting <= ending) {
    int n = ((ending - starting)/2) + 1;
    int ans = (n * (starting + ending)) /2;
    cout << ans << endl;
  }
 else {
    cout << 0 << endl;
  }
}
```

6. Three Numbers ;-

$X + Y + Z = S$ $\qquad 0 \le x, y, z \le k$

$X + Y = S - Z$

$X + Y = T$

Range of T: $S - K \le T \le S \simeq 0 \le T \le 2K$

Exa: $K = 8$

$X + Y = 7$

8 sol^n

$0 + 7$

$1 + 6$

$2 + 5$

$3 + 4$

$4 + 3$

$5 + 2$

$6 + 1$

$7 + 0$

$X + Y = 3$

$0 + 3$

$1 + 2$

$2 + 1$

$3 + 0$

→ 4 sol^n

$X + Y = 10$

$8 + 2$

$7 + 3$

$6 + 4$

$5 + 5$

$4 + 6$

$3 + 7$

$2 + 8$

→ 7 sol^n

$X + Y = 15$

$8 + 7$

$7 + 8$

→ 2 sol^n

$\underset{\text{sum}}{T} \le k$

↳ Then we are getting T+1 sol^n

When sum (T) > k ⇒ starting : $x \le k$, $\overset{\text{diff}}{y} = \overset{\text{sum (Given)}}{T} - k$

No. of sol^n = $K - \overset{\text{diff}}{\cancel{B}} + 1$

Exa:- $T = 10$, $k = 8$

diff $= 10 - 8 = 2$

No. of sol^n $= 8 - 2 + 1 = 7$

Code :-

T.C : O(K)

```
void solve ( ) {
   int k, s;
   cin >>k>>s;
   /*
    x+y +z =s;
    0 <=s<=3k
    x+y = s-z;
    x+y = T
    max (0, s-k) <=s-z<=s
   */
   int ans =0;
   for (int i= max ((int)0, s-k); i<=s; i++){
      if (i<=k){
         ans +=i+1;
      }
      else if (i<=2*k){
         int d =i-k;
         ans += (k-d+1);
      }
   }
   cout << ans << endl;
}
```

7. Another pyramid Problem :-

code :-

row = n

cal = (2 * n) - 1

space to print for every row = n - i

```
void solve (){
   int n;
   cin >>n;
   int row = n ;
   int cal = (2*n)-1
   for (int i=1; i<=row; i++){
     for (int blank=0; blank<n-i ; blank++){
        cout <<"*" ; }
     for (int num =1; num <=i; num++) cout << num <<" ";
     for (int num =i-1; num >=1; num --) cout << num << " ";
     for (int blank=0; blank <n-i; blank++){
        cout << "*"; }
     cout << endl;
}
```