Bit Manipulation

1. Left Shift

<<

10 << 1

0 1 0 1 0 = 10
1 0 1 0 0 = 20

(10 × 2 = 20) ⇒ Left Shift

$\boxed{a << b = a \times 2^b}$

2. Right Shift

0 0 1 1 0 0 = 12
0 0 0 1 1 0 = 6

$(a/2 \Rightarrow 12/2 = 6)$

3. XOR

0 ^ 0 = 0
1 ^ 1 = 0
0 ^ 1 = 1
1 ^ 0 = 1

(same number XOR → 0)

(~~To find the unique no.~~)

4. Check the no. even / odd (using bit manipulation) → fast

$2^3\ 2^2\ 2^1\ 2^0$

10 → 1010
6 → 0110

$$\begin{array}{r} 1010 \\ \&\ 0001 \\ \hline 0000 \end{array}$$ ~~odd~~ even

$$\begin{array}{r} 0110 \\ \&\ 0001 \\ \hline 0000 \end{array}$$ even

$\boxed{\begin{array}{ll} (n \& 1) == 0 & \text{even} \\ \text{else} & \text{odd} \end{array}}$

5. Check i^{th} bit is setbit or not

4^{th} part me LSB 0 hai ya 1 check kiya tha
5^{th} part generic hai (koi si bhi bit ke baare me pata karna hai)

Generic kaise banayenge?

⇓

i^{th} bit ko 1 ke saath AND karke

⇓

Mask

0 ① 0 1 1 0
0 $\boxed{1}$ 0 0 0 0

1
10
100
1000
10000

left shift 1 ($i-1$ times)

Code :-

```
for (int j=1; j<=i-1; j++){
        i = (i<<1);
 }
 n = n & i;
```

Much better way

⇓

```
public static void extractBit(int n, int i) {
        int j=1;                                   j → mask
        j = 1 << (i-1);
        n = n & j;      return (n==0? 0 : 1);
}
```

extractBit(6,5) → 0 → means i^{th} bit was zero
↘ 1 → means i^{th} bit was one

6. Set i^{th} bit

```
5 4 3 2 1 0
0 1 0 1 0 1
    ↑
  4th bit
```

4th bit set

OR with 1 ⟶ 1

```
  0 1 0 1 0 1
1 0 0 1 0 0 0
-------------
  0 1 1 1 0 1
```

1 << 3

```
1
10
100
1000
```

Code :-

```
public static int setBit(int n, int i) {
    int mask = 1;
    mask = 1 << (i-1);
    n = n | mask;
    return n;
}
```

4. ## Reset i^{th} bit

~~0 1 0 1 1 0~~ ~~i = 5~~

2^4 2^2 2^0

```
  0 0 1 0 1 0 1        3rd bit reset
& 1 1 1 1 0 1 1
---------------
  0 0 1 0 0 0 1
          ↑
```

0 0 0 0 1 0 0 → negation 1 1 1 1 0 1 1

Code :

```
public static int resetBit (int n, int i) {
        ~~int mask = 1;~~
        mask = ~(1 << (i-1)) ;
        n = n & mask ;
        return n ;
}
```

8. Posⁿ of rightmost set bit

$$\begin{array}{ccccccc} & 2^5 & 2^4 & & 2^2 & 2^1 & \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ & & & & & \uparrow & \\ & & & & & \underline{\underline{\text{Ans}}} & \end{array}$$

mask => 0000001

↓

upgraded mask => 0000010

Code :

```
public static int rightmostSetBit ( int n ) {

        int pos = 1 ;
        int mask = 1 ;

        while ( ( n & mask ) == 0 ) {
                pos ++ ;
                mask = mask << 1 ;

        }

        return pos ;

}
```

a ^ 1 = a

9) 9 + 1 ⇒ 10

↳ 01001 → 9

↓ HOW?

01010 → 10

first zero

HOW? ⇒ 0100111 → 39

0101000 → 40 Ans

```
      0100111
XOR         1 → mask
      -------
      0100110
XOR        10 → mask
      -------
      0100100
XOR       100 → mask
      -------
      0100000
         DONE!
         1000 → mask
      -------
      0101000
```

n & mask == 0
↳ stop

Code :

```
public static int addOne ( int n ) {
        int mask = 1 ;

        while(( n & mask ) != 0 ) {

            n = n ^ mask ;
            mask = ( mask << 1 ) ;

        }

        n = n ^ mask ;

        return n ;

}
```

10) Magic No. → no. which is power of 5 or sum of powers of 5
→ unique

5 ✓ , 10 ✗ $(5^1 + 5^1)$, 30 ✓ $(5^1 + 5^2)$

Tell i^{th} magic no. ?

Magic Nos. → 5, 25, 30, -------

II

$5^3 \ 5^2 \ 5^1$

$5 \rightarrow 1 \ 0 \ 1$

$\& \quad\quad 1$

$1 \ 0 \ 1 \rightarrow 5^1 = 5$ ✓

right ~~left~~ shift ⇒ $1 \ 0$
(drop LSB) $\& \ 1$
0 $\cancel{5^2}$

right ~~left~~ shift → 1
$\& \ 1$
$1 \rightarrow 5^3$ ✓

I base5 powers of 5 or sum of powers of 5

		5^3	5^2	5^1		
1	→	0	0	1	→	$5^1 = 5$
2	→	0	1	0	→	$5^2 = 25$
3	→	0	1	1	→	$5^2 + 5 = 30$
4	→	1	0	0	→	$5^3 = 125$
5	→	1	0	1	→	$5^3 + 5^1 = 130$
6	→	1	1	0	→	$5^3 + 5^2 = 125 + 25 = 150$
7	→	1	1	1	→	$5^3 + 5^2 + 5^1 = 155$

eg.
Toh 4 → ki binary repr^n ⇒ 1 0 0 → kahan kahan set bits hain
↓
wahan wahan 5 ki power ke acc sum krte jao

Code :

```
                                       → eg. 5
public static int magicNo (int i) {

        int pow = ~~8~~1; int ans = 0;
        while ( i! = 0) {                          5 → 101

              pow = pow * 5 ;
              if ((i & 1) ! = 0) {

                      ans = ans + pow;
              }
                i = (i >> 1);

        }
          return ans;

}
```

Dry run

5 != 0
pow = 5
ans = 5

2 != 0
pow = 25

bits

(10 >> 1 => 0)
1 != 0
ans = 5 + 5^3

⇓

130 => 5th
magical Number

Ans

(1 >> 1 => 0)

0 == 0
loop se bahar

11) Pascal's Triangle

I Give the sum of nth row

1st →	1						$\rightarrow 2^0 = 1$
2nd →	1	1					$\rightarrow 2^1 = 2$
3rd →	1	2	1				$\rightarrow 4 = 2^2$
4th →	1	3	3	1			$\rightarrow 8 = 2^3$
5th →	1	4	6	4	1		$\rightarrow 16 = 2^4$
6th →	1	5	10	10	5	1	$\rightarrow 32 = 2^5$

$$\boxed{n^{th} \text{ row} = 2^{n-1} \quad \text{Ans}}$$

II Sum of all the elements until nth row

e.g. 6th row

$$2^0 + 2^1 + 2^2 + \cdots + 2^5$$

↓

GP

$$2^{n+1} - 1 = 2^6 - 1$$

2^6 → 1 << 6

$$\boxed{\begin{array}{c} a << b \\ a * 2^b \end{array}}$$

Code :-

```
public static int pascalSum ( int n ) {
      int res = 1 << n
      return res -1;
}
```

12.) Check if number is power of 2 or not

8 ✓   9 X   16 ✓

$$\begin{array}{rl} 8 \rightarrow & 1000 \\ \& \; 7 \rightarrow & 0111 \\ \hline & 0000 \end{array}$$ } complement of 8

∴ 8 is a power of 2.

```
public static boolean powerOfTwo ( int n) {
        int res = n & (n-1) ;
        return ((res ==0)? true : false);

}
```

Special Case => $\begin{array}{r} 0 \\ -\,1 \\ \hline -1 \end{array}$ ⟶ 1 ⟶ 0001 } 2's comp.

1111

$$\begin{array}{rl} & 0000 \\ \& & 1111 \\ \hline & 0000 \end{array}$$

But 0 is not a power of 2.

Updated code

```
public static boolean powerOfTwo (int n) {
        if (n != 0) {
               if ((n & (n-1)) == 0) {
                        return true;
               }
        }
        return false;
}
```

13) Find missing two numbers

Given N & an array of size N-2 such that elements of the array will be of range 1 to N. We need to find those 2 missing numbers in O(n) & O(1)
time space

eg. N = 6
arr = {1, 3, 4, 6} → N-2 elements

$\boxed{\text{Ans} \rightarrow 2 \text{ and } 5}$

$$Z = \underbrace{\overset{\nearrow \text{arr}}{1 \wedge 3 \wedge 4 \wedge 6}} \wedge \underbrace{\overset{\nearrow \text{1 to N}}{1 \wedge 2 \wedge 3 \wedge 4 \wedge 5 \wedge 6}}$$

$Z = 2 \wedge 5$

```
2 -> 0 1 0
5 -> 1 0 1     ( 1^0 = 1 )
     -----
     1 1 1
       -> 7
```

$\boxed{Z = 7}$

```
        7
       / \
      2 ^ 5
      0   1
              = 1
      1   0
```

Right most set bit

```
  Z & ~(Z-1)
  7 & ~6
  7  => 1 1 1
  ~6 => & 0 0 1
        -------
          0 0 1      = 1
```

Z = 7 → $\boxed{1 1 \textcircled{1}}$

if it is 1 in either 2 or 5 & other bit should be zero

```
0 0 1 & 1
  0 0 1
& 0 0 1
-------
  0 0 (1)

2 => 0 1 0
   & 0 0 1
   -------
     0 0 (0)

3 => 0 1 1
     0 0 1
     -----
     0 0 (1)
```

① arr
② 1 – N
2 traversals

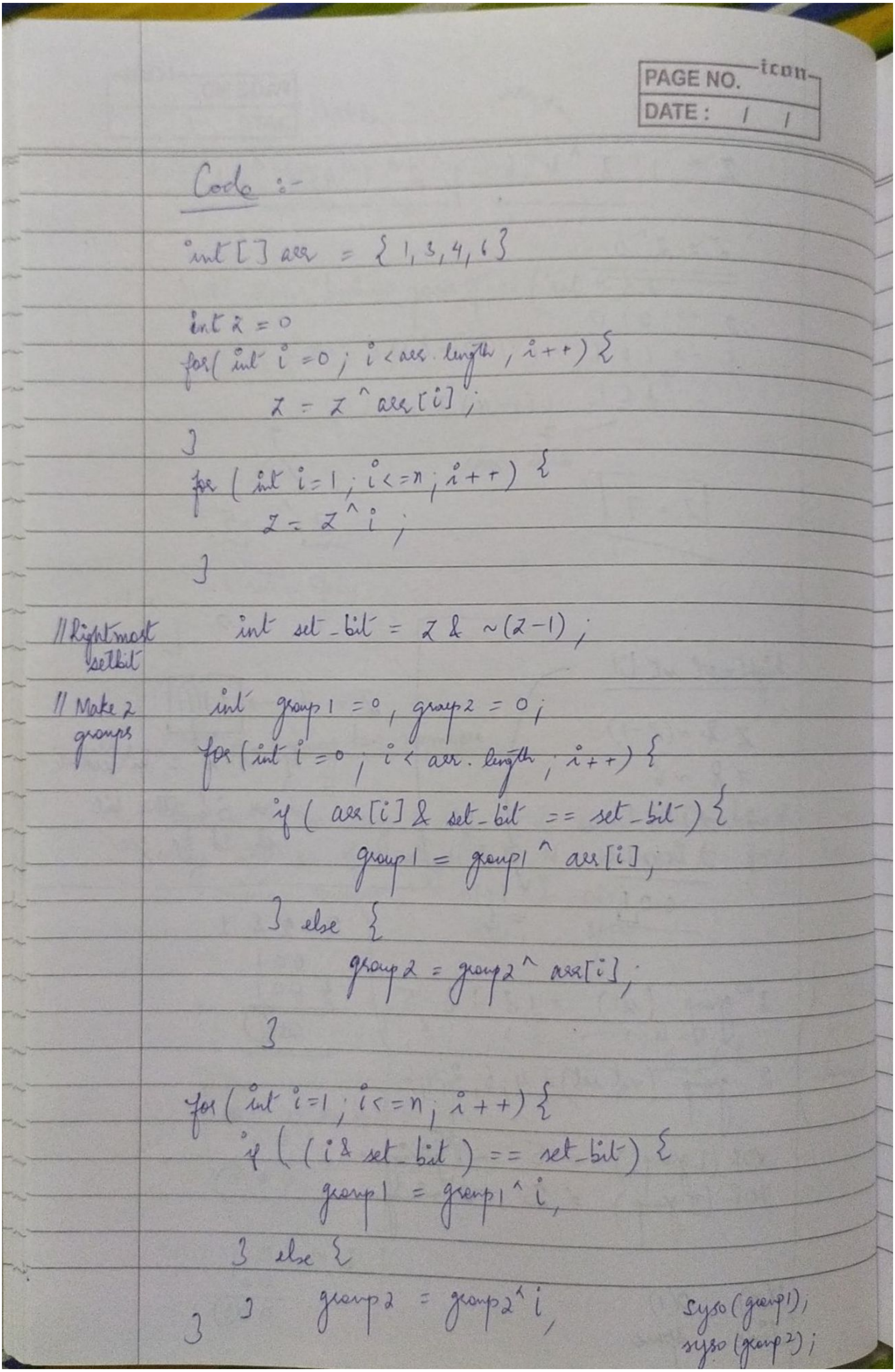
1st group (set) = 1, 3, 1, 3, 5

2nd group (not set) = 4, 6, 2, 4, 6

XOR (I group) = 5 — 1st missing no.

XOR (II group) = 2 — 2nd missing no.

O(n) Time   O(1) Space



Code :-

```
int [] arr = { 1, 3, 4, 6 }

int x = 0
for( int i = 0 ; i < arr.length , i++) {
        x = x ^ arr[i] ;
}
for ( int i = 1 ; i <= n ; i++ ) {
        x = x ^ i ;
}

//Rightmost setbit
        int set_bit = x & ~(x-1) ;

//Make 2 groups
        int group1 = 0, group2 = 0 ;
        for (int i = 0 ; i < arr.length ; i++) {
                if ( arr[i] & set_bit == set_bit ) {
                        group1 = group1 ^ arr[i] ;
                } else {
                        group2 = group2 ^ arr[i] ;
                }
        }

for ( int i=1 ; i<=n ; i++ ) {
        if ( ( i & set_bit ) == set_bit ) {
                group1 = group1 ^ i ;
        } else {
                group2 = group2 ^ i ;
        }
}
                                                syso(group1);
                                                syso (group2) ;
```