Number System

Bit Manipulation

① AND :-

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

★ When you 'and' any number with 1. The digits remain the same.

Eg:-

```
  1 1 0 0 1
& 1 1 1 1 1
-----------
  1 1 0 0 1   Same.
```

② OR :-

a	b	a OR b.
0	0	0
0	1	1
1	0	1
1	1	1

③ XOR (^) (if and only if)

Exclusive OR :-

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

OBSERVATIONS :-

1. $a \wedge 1 = \bar{a}$
2. $a \wedge 0 = a$
3. $a \wedge a = 0$

④ Complement (~)

$a = 10110$

$\bar{a} = 01001$

Number systems

1. Decimal → 0, 1, 2, ... 9 Base (10)

$(357)_{10}$, $(10)_{10}$

2. Binary → 0 & 1 Base (2)

$(10)_{10} = (1010)_2$

$(7)_{10} = (111)_2$

3. Octal → 0, 1, 2, 3..... 7 Base (8)

4. Hexadecimal → 0 1 2 3 4 5 6 7 8 9 A B C D E F

COMPARISON :-

DECIMAL :- 0 1 2 3 4 5 6 7 (8) 9 10 11

OCTAL :- 0 1 2 3 4 5 6 7 (10) 11 12 13

So, here 8 in decimal means 10 in octal.

$(9)_{10} = (11)_8$

HEXADECIMAL :- $(10)_{10} = (A)_{16}$.

$(10)_8 = (8)_{16}$

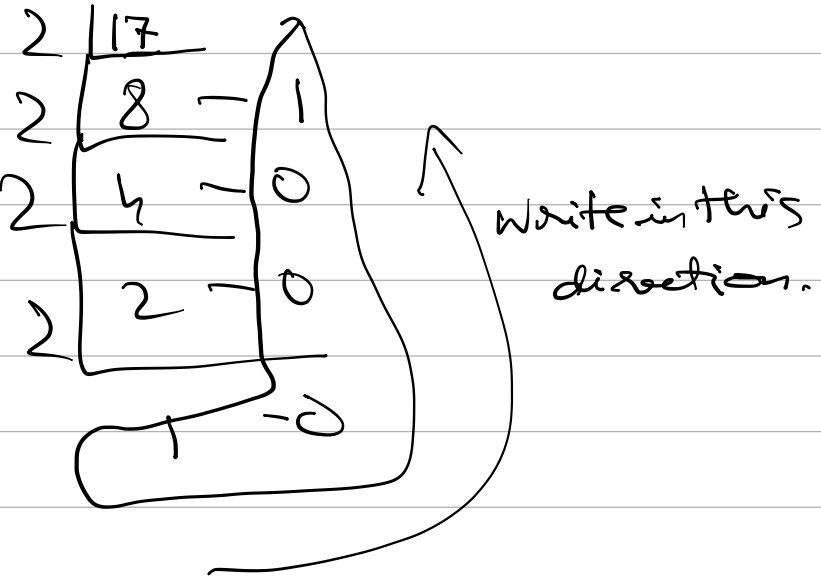
Conversions

1. Decimal to base b.
2. base b to Decimal.

1. Decimal to base b :-

Q). Convert $(17)_{10}$ to base 2.

Keep dividing by base, take remainders, write in opposite.



$(10001)_2 = (17)_{10}$

Q). $(17)_{10} = (?)_8$.

$8 \mid 17$

$2 - 1$

Write in opposite.

$(17)_{10} = (21)_8$.

(2) Convert any base b to decimal:-

Q). $(10001)_2 = (?)_{10}$.

Steps:- Multiple and add the power of base with digits.

$= 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

$= 16 + 0 + 0 + 0 + 1 = (17)_{10}$

Q). $(21)_8 = (?)_{10}$.

$= 8 \times 2 + 8^0 \times 1 = (17)_{10}$

NOTE:- If the question is to convert base a number to base b number.

*). First convert base a to decimal.

*). Next convert decimal to base b.

---

(5)- Left shift operator (<<) ("used to double a number")

Step-1:- $(10)_{10} = (1010)_2$

Convert any number to binary first.

Step-2 :- $1010 << 1 = 10100$

$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

$= 20$

$\therefore a << 1 = 2a$

General Point:-

$\therefore$ $a << b = a \times 2^b$

6]. Right Shift :- >>

Q). 00110001 >> 1

Solu:- => 00110 0

GENERAL POINT:-

$$a >> b = a \times \frac{1}{2^b}$$

Questions

1]. Given a number n. Find whether it is odd or even.

Solu:-

Logic :- If 2^0 place is 1 then number is odd else it is even.

∴ If n & 1 == 1 => odd else even.

```java
public class OddEven {

    public static void main(String[] args) {
        int n = 68;
        System.out.println(isOdd(n));
    }

    private static boolean isOdd(int n) {
        return (n & 1) == 1;
    }
}
```

Output is :- false as 68 is even and 68 & 1 will return 0.

=

---

(2). An array is given. ,every number appears twice ,but only one number appears once.
Find that number.

Solu:- Logic:- XOR all elements in the array. We know, $a \wedge a = 0$.
So, if the duplicate element is present, it will become 0.

```java
public class FindUnique {
    public static void main(String[] args) {
        int[] arr = {2, 3, 3, 4, 2, 6, 4};
        System.out.println(ans(arr));
    }

    private static int ans(int[] arr) {
        int unique = 0;

        for(int n : arr) {
            unique ^= n;
        }

        return unique;
    }
}
```

Output:-

6.

=

Q]. Find i^{th} bit of a number.

1 0 1 1 0

6 5 4 3 2 1

Solution: 1 1 0 1 1 0

Suppose we have to find 3rd bit.

1 1 0 1 1 0

& 0 0 0 1 0 0 → mask.

0 0 0 1 0 0

Logic:- We use left shift to get this number.

To get 2 zeroes on the right side of 1 we use,

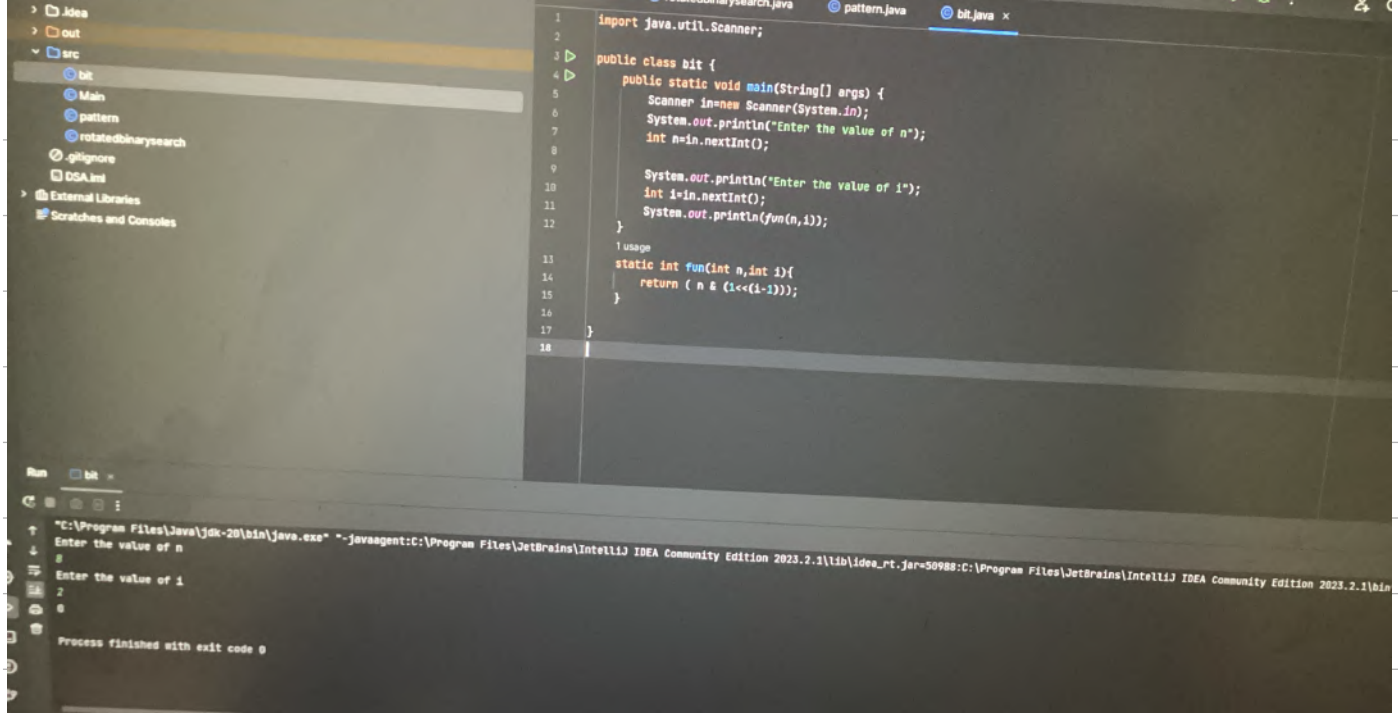
1 << 2 (here 3rd bit).

i.e 1 << (i-1)

(i -> bit to find).

Later we 'AND' this 1<<(i-1) to number.

n & (1<< (i-1))



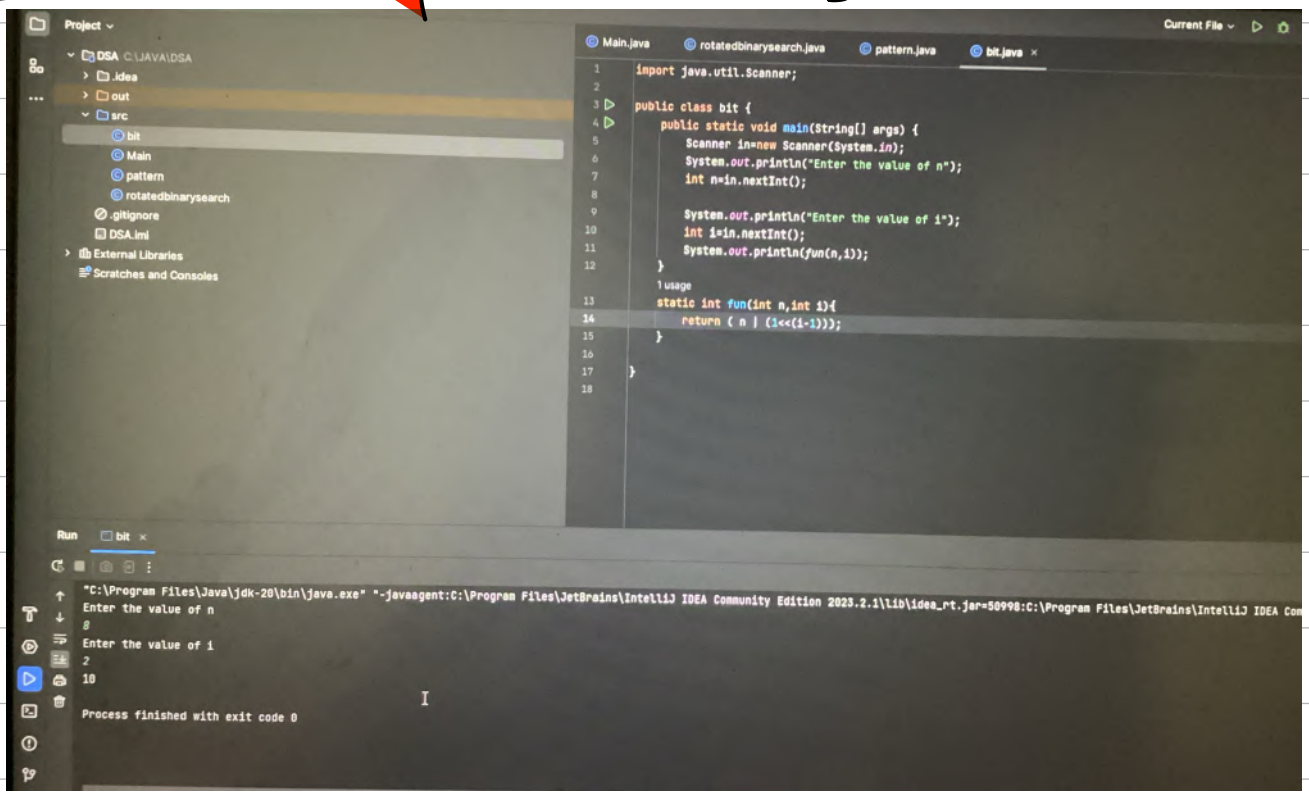
Q). Set i^{th} bit to 1.

Soln:-

$$\begin{array}{r} 10\ 10\ 11\ 0 \\ (OR)\ 00\ 00\ 1\ 000 \rightarrow \text{Mask.} \\ \hline 10\ 10\ 1\ 11\ 0 \end{array}$$

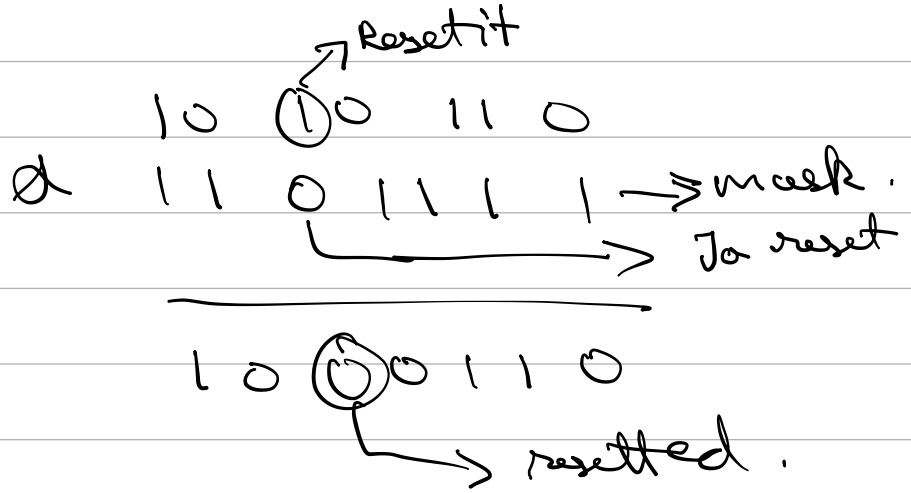
Here 8 → 1000
2nd bit is set
So, 1010 = 10

set to 1.

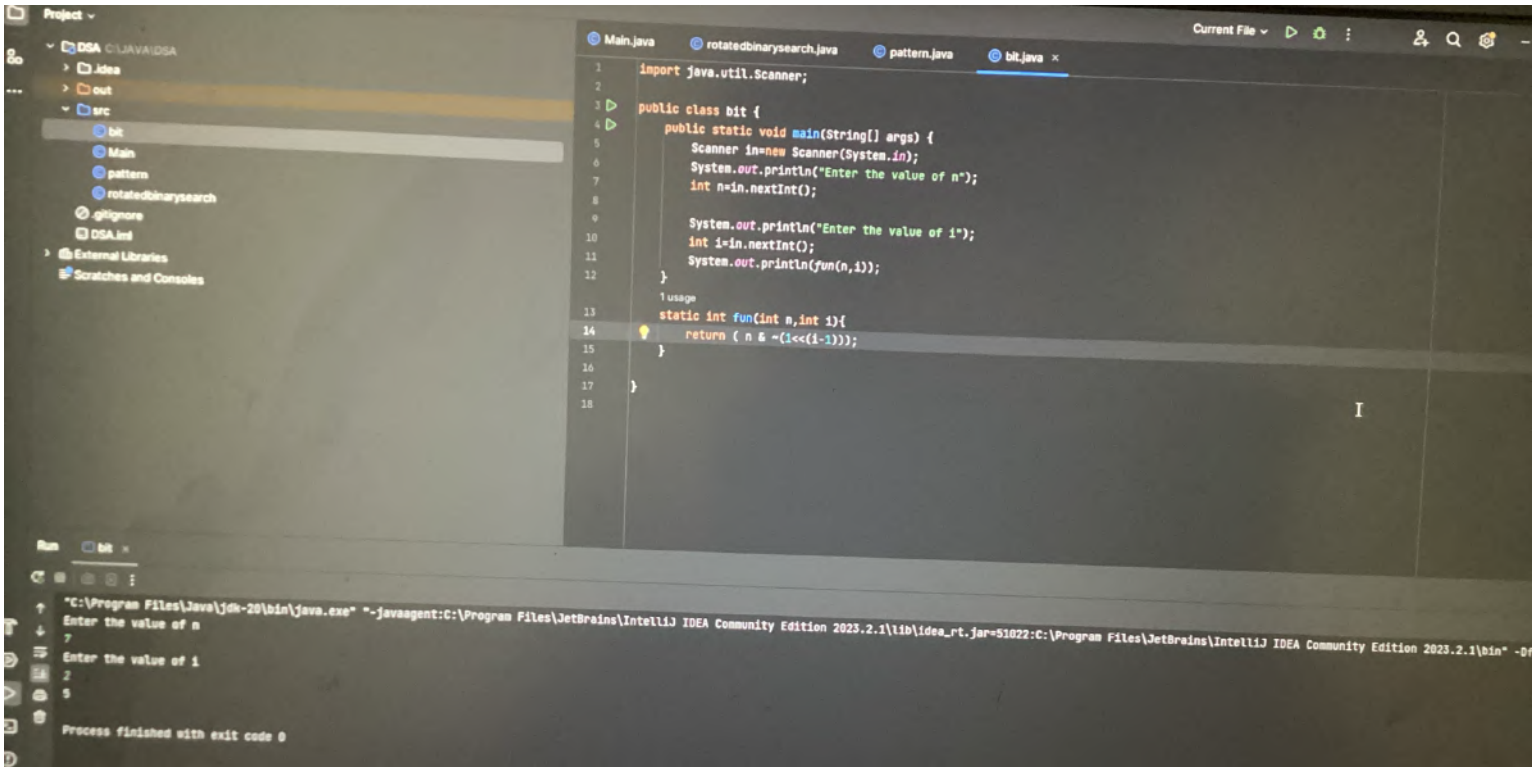


Q). Reset i^{th} bit :-

Soln :-



$n \;\&\; \sim(1 << (i-1))$



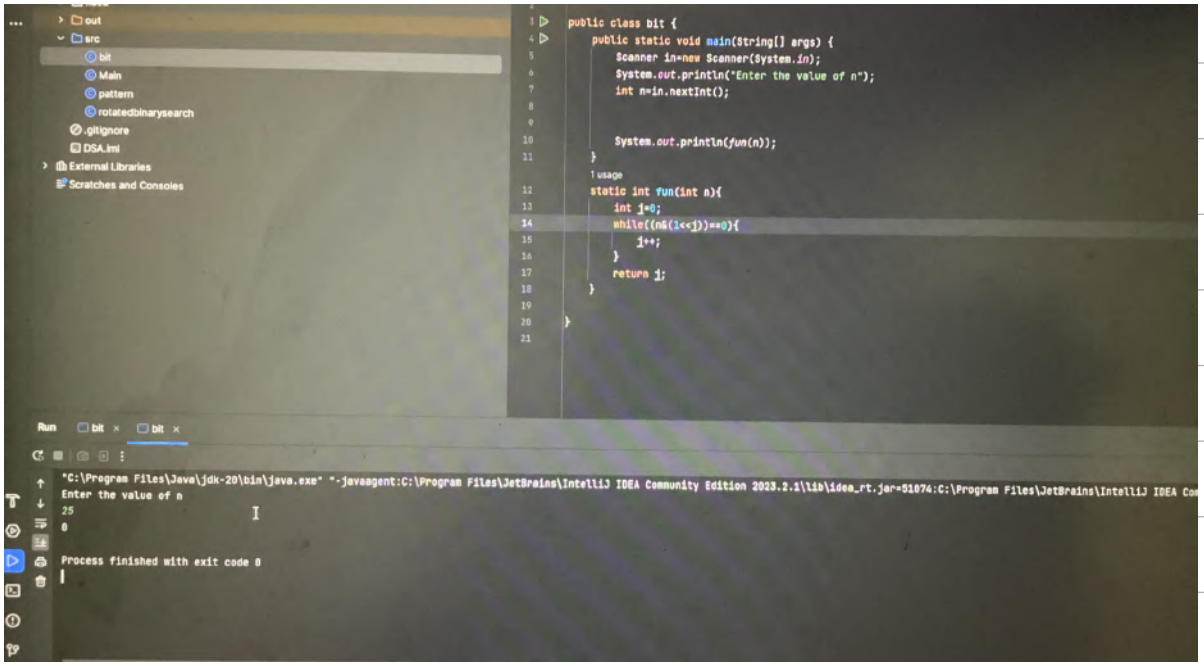
Q). Find the position of the rightmost set bit.

Soln :-

1 1 0

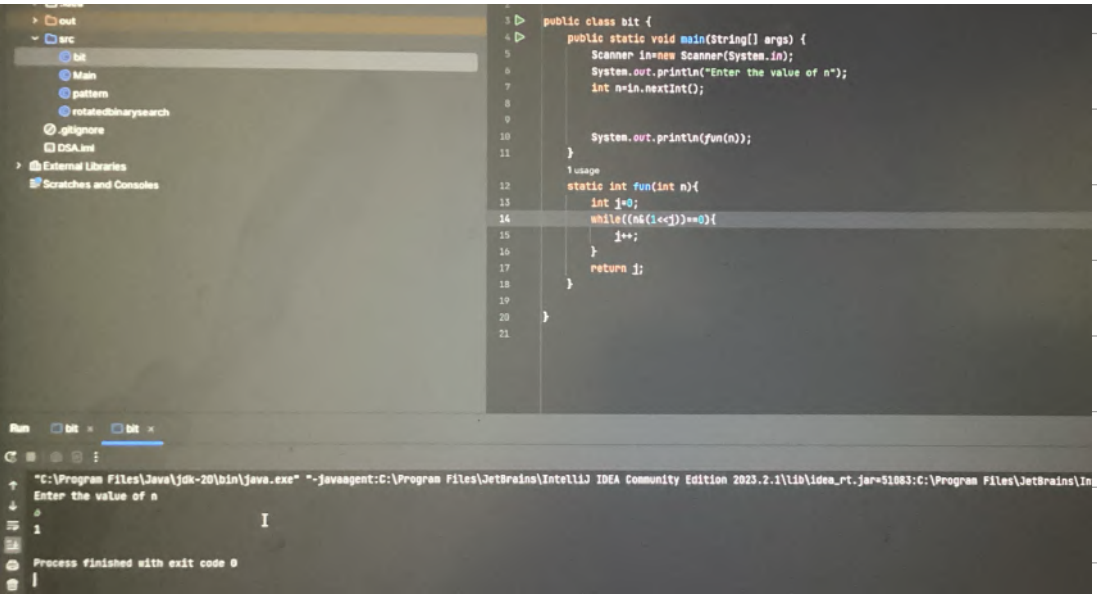
i = 0;

```
while((n & (1<<i)) != 0)
{
    i++;
}.
```



Here 25, is

0 1 1 0 0 (1) → 0th position



Here 6, is

1 (0) → 1st position.

So 1

SIMPLE FORMULA WITHOUT USING WHILE LOOP IS.

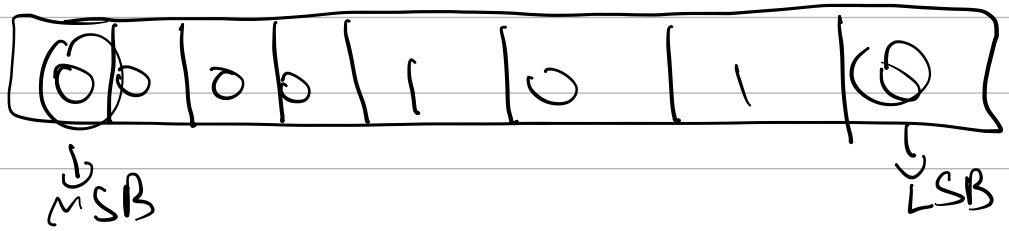
$N \& (-N)$.

FOR THIS WE WILL LEARN NEGATIVE BINARY NUMBERS :-

Negative of a number in Binary form

1 byte := 8 bits.

10 =>



MSB -> Tells us whether the number is +ve or -ve

1 -> -ve

0 -> +ve.

LSB -> Tells whether the number is even or odd.

Step-1:- (1) Take complement of no.

(2). Add 1 to it.

$(10)_{10} = (00001010)_2$

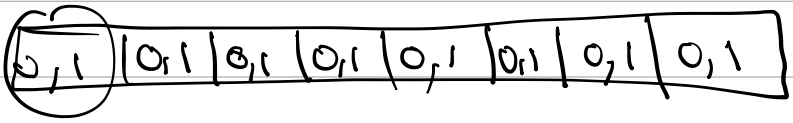
(1) (1111 0101)

(2).
$$\begin{array}{r} 1111\,0101 \\ +\quad\quad 1 \\ \hline 1111\,0110 \end{array}$$

$(1111\,0110)_2 \Rightarrow (-10)$.

Range of numbers

(1) 1 byte :-



Total = $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$

$= 2^8 = 256$ numbers

Actual number stored in bit = $n-1$

n → ~~total~~ bits.

i.e 1 byte = 7 bits. (1 bit for sign).

means $2^7 = 128$ numbers

When MSB = 1 , −128 numbers
MSB = 0 , 128 numbers.

from −128 to 127 there are 257 numbers (128+128 − 1 (0 is in middle))

Range of Number

-2^{n-1} to $2^{n-1} - 1$

n → bits.

FORMULA

Questions Continued

Q8]. All numbers are repeating 3 times, only one number is appearing once, Find that time.

arr = [2, 2, 3, 2, 7, 7, 8, 7, 8, 8]

Soln:- Logic :- The numbers which are appearing 3 times, have their bit positions appearing 3 times as well. If we add all the numbers and take the modulo, we get the extra number.

```
  1 0
  1 0
  1 1
  1 0
1 1 1
1 1 1
1 0 0 0
  1 1 1
1 0 0 0
1 0 0 0
-------
3 3 7 4
```

↳ There is extra 1 1 → means answer is 3.

Q9]. Find the n^{th} magic number.

Amazon Question

	5^3	5^2	5^1		Magic no
1 =	0	0	1	→	5
2 =	0	1	0	→	10
3 =	0	1	1	→	30
4 =	1	0	0	→	125

:

n

Solu: If n is given. n^{th} magic number is

Loop.

- n & 1 => This will give me last digit in binary.
- n >> 1 => To proceed to next number.

Till n=0 number becomes 0.

```java
public class MagicNumber {
    public static void main(String[] args) {
        int n = 6;

        int ans = 0;
        int base = 5;
        while (n > 0) {
            int last = n & 1;
            n = n >> 1;
            ans += last * base;
            base = base * 5;
        }

        System.out.println(ans);
    }
```

Q]. Find number of digits in base b.

Soln $(6)_{10} = 1$ → if base = 10, divide by 10 and increment counter till n=0.

$(6)_{10} = (110)_2 = 3.$

↓

use right shift till n=0 increment the counter.

Other than these two methods,

General formula :-

$= \text{int}(\log_b n) + 1$

n → any number

b → at which base the number is given.

Eg:- $(6)_2 \rightarrow b$ (n)

$= \text{int}(\log_2 6) + 1$

$= 2 + 1 = 3$ (correct)

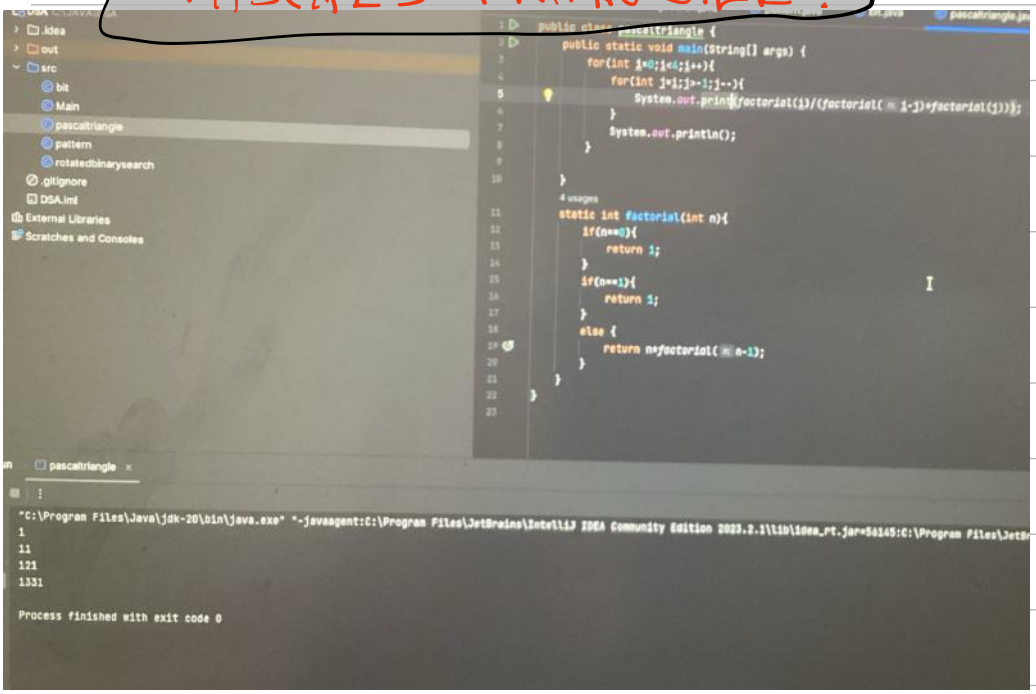
110

```java
public class NoOfDigits {
    public static void main(String[] args) {
        int n = 34567;
        int b = 10;

        int ans = (int)(Math.log(n) / Math.log(b)) + 1;

        System.out.println(ans);
    }
}
```

To find $\log_b a$

$$\log_b a = \frac{\log_{10} a}{\log_{10} b}$$

PASCAL'S TRIANGLE :-




```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

logic:- Each element position is given by.

${}^{n}C_{a}$.

n => row; 0, 1, 2, ...

a = column; 0, 1, 2, ...

Q]. Find the sum of n^{th} row :-

Sol:- Sum of each row = $n = 0, 1, 2, 3, \ldots$

$${}^nC_0 + {}^nC_1 + {}^nC_2 + \ldots\ldots + {}^nC_n = 2^n$$

For n^{th} row, sum $= 2^{n-1}$

Answer:- $1 << (n-1)$

$\Rightarrow 1 \times 2^{n-1}$, $n = 1, 2, 3 \ldots\ldots$

Q). You are given a number, find out if its power of 2 or not.

Eg:- 16 → 10000 → Power of two

20 → 10100 → more than one 1's are there so, not power of 2.

Generic method, keep right shifting, increment counter when 1 is encountered;

$n \& 1 = 1$

if counter > 1, not a power of 2.

Formula :- We know if a number n is given and we AND it with number -1, we should get 0, if it is power of 2.

Eg :-

$1000 = 8$ (power of 2)

$\&\ 111 = 7$ ($n-1$ i.e $8-1$)

$0000 = 0$ (so 8 is power of 2).

`(n & (n-1)) == 0;`

↓

returns true if it is power of 2

```
public class PowOfTwo {
    public static void main(String[] args) {
        int n = 32;
        boolean ans = (n & (n-1)) == 0;
        System.out.println(ans);
    }
}
```

→ Returns true as 32 is power of 2.

Q). Calculate a^b.

$3^6 = 3 \times 3 \times 3 \times 3 \times 3 \times 3$.

Time complexity = O(b).

Instead use,

$3^{(110)_2}$ concept.

Logic :- Using right shifting of power and multiplying with base (initially 3) when power & 1 == 1 condition is true and ignore when power & 1 == 0, but base = base x base always.

```java
public class Power {
    public static void main(String[] args) {
        int base = 3;
        int power = 6;

        int ans = 1;

        while (power > 0) {
            if ((power & 1) == 1) {
                ans *= base;
            }

            base *= base;
            power = power >> 1;
        }

        System.out.println(ans);
    }
}
```

Q]. Given a number, find the number of set bits in it.

Solu:-

Generic method,

Do, n & 1

If n & 1 == 1, counter ++,

n >> 1

Till n == 0. (i.e n > 0)

Formula:-

We know, to get rightmost set bit, we used, N & (-N)

Eg:- 1001

N & (-N) => 0001

So, if we do N - [N & (-N)] = 1000

counter ++;

i.e loop until n > 0,

counter gives no of set bits.

No of set bits = No of iterations.

(OR)

```
while (n>0) {
    counter++
    n=n & (n-1);
}
```

Also correct.

For example:- n = 1010 = 10

n-1 = & 1001 = 9.

Now n = 1000 = 8

n-1 = 0111 = 7

n = 0000

n=0 terminate.

counter = 2.

```
11      private static int setBits(int n) {
12          int count = 0;
13
14  //          while (n > 0) {
15  //              count++;
16  //              n -= (n & -n);
17  //          }
18
19          while (n > 0) {
20              count++;
21              n = n & (n-1);
22          }
23
24          return count;
```

Q). Find XOR of numbers from 0 to a.

Solⁿ:

a	XOR from 0 to a
0	0
1	0^1 = 1
2	0^1^2 = 0^1^10 = 11 = 3
3	0^1^2^3 = 11^11 = 0
4	0^1^2^3^4 = 0^4 = 4
5	1
6	7
7	0
8	8
9	1

Observe:-

If $(a \% 4 == 0)$ XOR = a.

$a \% 4 == 1$ XOR = 1

a % 4 == 2
XOR = a+1

a % 4 == 3
XOR = 0.

↓

Loop this from 0 to a.

Q]. XOR of all numbers between a & b.

Soln:- If a = 3 & b = 9.

3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8 ^ 9.

We already studied 0 to a.
so, to find a to b.

Do, 0 to b − 0 to a-1

0 ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8 ^ 9
− 0 ^ 1 ^ 2

NOTE:- ^ (XOR) operation in binary is equivalent to subtraction.

So Logic:- ans :- XOR(b) ^ XOR(a-1)
till b from 0 to a-1
from 0

```java
public class RangeXOR {
    public static void main(String[] args) {
        // range xor for a, b = xor(b) ^ xor(a-1)
        int a = 3;
        int b = 9;

        int ans = xor(b) ^ xor(a: a-1);

        System.out.println(ans);
```

```java
// this gives xor from 0 to a
static int xor(int a) {
    if (a % 4 == 0) {
        return a;
    }

    if (a % 4 == 1) {
        return 1;
    }

    if (a % 4 == 2) {
        return a + 1;
    }

    return 0;
}
```

Maths for DSA - II

Next Notes