**BIT Manipulation-**

**1. Square of Number🡪**

public static int square(int num)

    {

        if (num < 0)

            num = -num;

        int power = 0, result = 0;

        int temp = num;

        while (temp > 0) {

            if ((temp & 1) > 0) {

                result += (num << power);

            }

            power++;

            temp = temp >> 1;

        }

        return result;

    }

**2. Single Number🡪**

public int[] singleNumber(int[] nums) {

        int xor = 0;

        for (int num : nums) {

            xor ^= num;

        }

        int rightmostSetBit = xor & -xor;

        int num1 = 0;

        int num2 = 0;

        for (int num : nums) {

            if ((num & rightmostSetBit) == 0) {

                num1 ^= num;

            } else {

                num2 ^= num;

            }

        }

        if (num1 > num2) {

            int temp = num1;

            num1 = num2;

            num2 = temp;

        }

        return new int[] { num1, num2 };

    }

**3. SETBIT COUNT🡪**

static int setBits(int N) {

        // code here

        return Integer.bitCount(N);

    }

    public static int countSetBits(int n) {

        // Write your code here.

        int res = 0;

        while (n != 0) {

            res += n & 1;

            n >>= 1;

        }

        return res;

    }

**4. Is POWER of 2🡪**

public static boolean isPowerofTwo(long n){

        if(n<=0){

            return false;

        }

        return (n&(n-1))==0;

    }

    public static boolean isPowerofTwo2(long n){

        if(n==0){

            return false;

        }

        while(n%2==0)

        {

            n/=2;

        }

        if(n==1){

            return true;

        }

        else{

            return false;

        }

    }

**5. Find SETBIT Position🡪**

static int findPosition(int N) {

        int pos=0;

        int i=0;

       while(N!=0){

            if((N&1)==1)

                pos++;

            i++;

            N>>=1;

        }

        if(pos==1){

            return i;

        }

        return -1;

    }

**6. Counts bits Flip🡪**

public static int countBitsFlip(int a, int b){

        // Your code here

        int cnt=0;

        int res=a^b;

        while(res!=0){

            cnt+=res&1;

            res>>=1;

        }

        return cnt;

    }

    public static int countSetBits(int n){

        // Your code here

        if(n==0){

            return 0;

        }

        int x=largerPowerOf2(n);

        int y=x\*(1<<(x-1));

        int z=n-(1<<x);

        return y+z+1+countSetBits(z);

    }

    public static int largerPowerOf2(int n){

        int x=0;

        while((1<<x)<=n){

            x++;

        }

        return x-1;

    }

**7. Copy Set Bits🡪**

 static int copySetBits(int x, int y, int l, int r) {

        if (l < 1 || r > 32)

            return x;

        for (int i = l; i <= r; i++) {

            int mask = 1 << (i - 1);

            if ((y & mask) != 0)

                x = x | mask;

        }

        return x;

    }

**8. PositionSetResetValue🡪**

class PositionSetResetValue {

    public static void naiveApproach() {

        int n=11;

        int x=2;

        int pos=0;

        int ans1=0;

        int ans2=0;

        while(n>0){

            x--;

            int d=n&1;

            if(d==1 && x!=0){

                ans1+=1<<pos;

                ans2+=1<<pos;

            }

            if(x==0){

                System.out.println(d);

                ans1+=1<<pos;

            }

            n>>=1;

            pos++;

        }

        System.out.println(ans1+" "+ans2);

    }

    public static void simpleApproach() {

            int n=11;

            int x=2;

            int get=0;

            int set=0;

            int clear=0;

            int mask=1<<(x-1);

            if((n&mask)!=0){

                get=1;

            }

            set=n|mask;

            clear=n&(~mask);

            System.out.println(get+" "+set+" "+clear);

    }

}

**9. Number Setting RightMost UnsetBit🡪**

public class NumberSettingRightMostUnsetBIt {

    public static int setBits(int N){

        // Write your code here.

        if((N&(N+1))==0){

            return  N;

        }

        return  N|(N+1);

    }

}

**10. Kth SetBIT🡪**

public class isKthBitSet {

    static boolean isKthBitSet(int n, int k) {

        return ((n >> (k - 1)) & 1) == 1;

    }

}

**11.Power(x,n)🡪**

public class powerOfNumber {

    public double myPow(double x, int n) {

        if (n < 0) {

            n = -n;

            x = 1 / x;

        }

        double pow = 1;

        while (n != 0) {

            if ((n & 1) != 0)

                pow \*= x;

            x \*= x;

            n >>>= 1;

        }

        return pow;

    }

}

**12. Check ith bit Set OR Not🡪**

class CheckIthBitSet {

    public static void main(String[] args) {

        int n = 4;

        int i = 3;

        int m = 1 << (i - 1);

        if ((m & n) != 0) {

            System.out.println("Yes");

        } else {

            System.out.println("No");

        }

    }

}

**13. Set ith bit🡪**

class SetIthBit {

    public static void main(String[] args) {

        int n = 4;

        int i = 4;

        int m = 1 << (i - 1);

        n=n|m;

        System.out.println(n);

    }

}

**14. Delete ith bit🡪**

class DeleteIthBit {

    public static void main(String[] args) {

        int n = 4;

        int i = 3;

        int m = 1 << (i - 1);

        n=n&~m;

        System.out.println(n);

    }

}

**15. Delete Last Set Bit**

class DeleteLastSetBit {

    public static void main(String[] args) {

        int n = 10;

        int i = 3;

        n=n&(n-1);

        System.out.println(n);

    }

}

**16. Toggle ith Bit of Number🡪**

class ToggleIthBit {

    public static void main(String[] args) {

        int n = 10;

        int i = 3;

        int m=1<<(i-1);

        n=n^m;

        System.out.println(n);

    }

}