***BIT MANIPULATION***

***//Swapping 2 number without third varibale***

*int* a=5;

*int* b=6;

 a=a^b;

 b=a^b;

a=a^b;

*static* *void* **checkithbit**(*int* *n*,*int* *i*){

        if((n & (1<<i))==0){

            System.out.println(" ith not set");

        }

        else{

            System.out.println("ith Set");

        }

    }

*static* *int* **setting\_ith\_bit**(*int* *n* ,*int* *i*){

        return (n | (1<<i));

    }

*static* *int* **clearing\_ith\_bit**(*int* *n* ,*int* *i*){

        return (n & (~(1<<i)));

    }

*static* *int* **remove\_last\_setbit**(*int* *n*){

        return n&(n-1);

    }

*static* ***boolean* is**\_power\_of2(*int* *n*){

       return n > 0 && (n & (n - 1)) == 0;

    }

*static* *int* **number\_of\_setBits**(*int* *n*){

*int* count=0;

        while(n!=0){

            n= (n&n-1);

            count++;

        }

        return count;

    }

*static* *String* **DtoB**(*int* *dec*){

*StringBuilder* result = new StringBuilder();

        while(dec>0){

            result.append(dec%2==0?'0':'1');

            dec/=2;

        }

*String* str ="";

        for(*int* i=result.length()-1 ;i>=0;i--){

            str+=result.charAt(i);

        }

        return str;

    }

*static* *int* **BtoD**(*String* *Bin*){

*int* result=0 ;

*int* pow=1;

        for(*int* j=Bin.length()-1;j>=0;j--){

            if(Bin.charAt(j)=='1'){

                result+=pow;

            }

            pow\*=2;

        }

        return result;

    }

**//Minimum Number of bits to create new number**

*static* *int* count(*int* *num1*,*int* *num2*){

*int* disitinctbitsetto1 =num1^num2;

*int* count=0;

*//count the set bits*

        while (disitinctbitsetto1!=0){

            count++;

            disitinctbitsetto1=disitinctbitsetto1&(disitinctbitsetto1-1);

        }

        return count;

    }

**//Power Set**

*static* *void* **powersetsolve**(*String* *str*,*List*<*String*> *result*){

        for(*int* i =0 ;i<(1<<str.length());i++){

*String* tempstr="";

            for(*int* j =0 ;j<str.length();j++){

                if((i&(1<<j))!=0){

                    tempstr+=str.charAt(j);

                }

            }

            result.add(tempstr);

        }

    }

**//SINGLE NUBMER 1 2 3**

***//Single number 1 (pair off two and a single to find)***

*static* *int* find1(*int* *arr*[]){

*int* result=0;

        for(*int* i :arr){

            result^=i;

        }

        return result;

   )

***//single number 2 (pairs of 3 an done single to find)***

*static* *int* find(*int* *arr*[]){

*int* ones=0 ,twos=0;

        for(*int* i : arr){

            ones= (ones^i)&(~twos);

            twos= (twos^i)&(~ones);

        }

        return ones;

    }

***//single number 3 (pairs of 2 and two singles to find)***

*static* *void* find3(*int* *arr*[]){

*int* result=0;

        for(*int* i :arr){

            result^=i;

        }

*int* firstbitset = (result&result-1)^result;

*int* num1 =0;

*int* num2=0;

        for(*int* i :arr){

            if((i ^ firstbitset) == 0){

                num2^=i;

            }else{

                num1^=i;

            }

        }

        System.out.println(num1);

        System.out.println(num2);

    }

**//XOR ing a rnge without XOR one by one**

*// 1 3 0 4 , 1 7 0 8 ,1 11 0 12*

*static* *int* findXor(*int* *n* ){

        if(n%4==1)return 1;

        else if(n%4==2) return n+1;

        else if(n%4==3) return 0;

        else return n;

    }

*static* *int* xorRange (*int* *l* ,*int* *r*){

*int* \_1toL = findXor(l);

*int* \_1toR =findXor(r);

        return \_1toL^\_1toR ;

**//Dividing without division or multiplication operator**

*static* *int* divide(*int* *num* ,*int* *div*){

        if (div == 0) return 0 ;

        if (num == Integer.MIN\_VALUE && div == -1) return Integer .MAX\_VALUE;

*boolean* negative = false;

        if(num<0 && div>0) negative =true;

        if(num>0 && div<0) negative= true;

*long* numt= Math.abs(num);

*long* divt =Math.abs(div);

*//keep subtracting mac div \* power of 2 till possible*

*int* result = 0;

        while(num >=div){

*int* power =0;

            while(num>=(div<<power)){

                power++;

            }

            num = num-(div<<(power-1));

            result +=(1<<power-1);

        }

        return negative ? - result : result;

    }