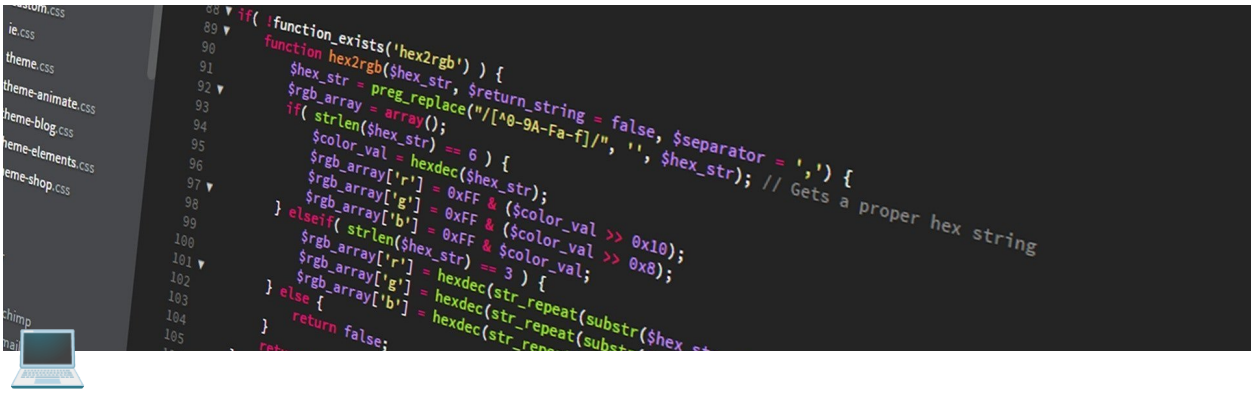


Kunal Kushwaha DSA Bootcamp

NOTES -

Q. Code for checking the occurrence of a digit in a number .

```
import java.util.Scanner;

public class Occurence {
    public static void main(String[] args) {
        System.out.println("enter the number :");
        Scanner sc =  new Scanner(System.in);
        int num = sc.nextInt();
        System.out.println("enter the digit you want to check:");
        int digit = sc.nextInt();
        int count= 0;
         sc.close();
        while(num>0){
            int rem = num%10;     // gives the last digit
            if (rem==digit){
                count++;
            }
            num/=10;            // gives the number after discarding the last digit
        }
        System.out.println(" No of times "+ digit+ " occured is "+ count);
    }
}
```

2. Code to reverse a given number.

```
import java.util.Scanner;

public class reverse {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number: ");
        int num = sc.nextInt();
        String s=String.valueOf(num);  //converts int into string
        int power = s.length();        //used to find out the length of the string;total no of digit that would later be used as powers of
        double rev = 0;                // stores the reverse number of abc..yz i.e. z*10**n + y*10**(n-1) + x *10 ** (n-2) + ....
        sc.close();
        while(power>0){
            int rem = num%10;         // extracting the last digit
            rev  = rev + rem*Math.pow(10,power-1) ;  //reverse number ;
            power -- ;
            num/=10;
```

```
        }
        System.out.println("reverse of the given number is :" + rev);
    }
}
```

OR

```
import java.util.Scanner;

public class reverse {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number: ");
        int num = sc.nextInt();
        int rev = 0;
        sc.close();
        while(num>0){
            int rem = num%10;
            num/=10;
            rev = rev *10+rem;
        }
        System.out.println("the reverse of the given number is: "+ rev);
    }
}
```

Assignment - 03-conditionals-loops.md

1. Take integer inputs till the user enters 0 and print the sum of all numbers.

```
import java.util.Scanner;
public class demo
{

public static void main(String args[])
{
System.out.println("enter the numbers: ");
Scanner sc = new Scanner(System.in);
int sum = 0;
while(true){
    int num = sc.nextInt();
    sum = sum+num;
    if (num==0){
        break;
    }
  }
 sc.close();
System.out.println("the sum of the numbers is "+ sum);
 }
}
```

2. Take integer inputs till the user enters 0 and print the largest number from all.

```
import java.util.Scanner;
public class demo
{

public static void main(String args[])
{
System.out.println("enter the numbers: ");
```

```
Scanner sc = new Scanner(System.in);
int max = 0;
while(true){
    int num = sc.nextInt();
    if(num>max){
        max=num;
    }
    if (num==0){
        break;
    }
  }
 sc.close();
System.out.println("the largest number is "+ max);
 }
}
```

3. Factorial

```
import java.util.Scanner;
public class demo
{

public static void main(String args[])
{
System.out.println("enter the numbers: ");
Scanner sc = new Scanner(System.in);
int num = sc.nextInt();
int factorial = num;
for (int i=num-1; i>0;i--){
factorial = factorial * i;
}
System.out.println("factorial of "+ num + " is "+ factorial);
 }
}
```

4. Display average of numbers when input is 0 .

```
import java.util.Scanner;
public class demo
{

public static void main(String args[])
{
System.out.println("enter the numbers: ");
Scanner sc = new Scanner(System.in);
double sum = 0;
double avg = 0;
int count = 0;
while(true){
int num = sc.nextInt();
if (num==0){
    break;
}
sum = sum+num;
count++;
}
avg = sum / count;
System.out.println(" the average of the numbers is "+ avg);
 }
}
```

5. Subtraction of product and sum of digits of a number

```
import java.util.Scanner;
public class demo
{

public static void main(String args[])
{
System.out.println("enter the number: ");
Scanner sc = new Scanner(System.in);
int num  = sc.nextInt();
int temp  =num;
int mul = 1;
int rem  = 0;
int sum = 0;
   while(num>0){
    rem  = num%10;
    sum = sum+rem;
    mul  = rem*mul;
    num/=10;
    }
  System.out.println("difference between product and sum of digits of " + temp + " is : " +(mul-sum));
 }
}
```

6. Permutation and combination

```
import java.util.Scanner;
public class demo
{
// permutation and combination
public static void main(String args[])
{
System.out.println("Enter n : ");
Scanner sc = new Scanner(System.in);
int n  = sc.nextInt();
System.out.println("Enter r:");
int r = sc.nextInt() ;
int a  =1;
int b = 1;
int c  =1;
 for (int i = n; i>0 ; i--){   //n!
   a = a*i;
   }
 for (int i = n-r ; i>0 ; i--){  //(n-r)!
    b = b*i;
    }
    System.out.println("P(n,r) is : "+ a/b);


 //  combination


 for (int i= r; i>0 ;i-- ){ //r!
  c = c*i;
 }
System.out.println("C(n,r) is : "+ a/(c*b));
  }
}
```

7. LCM

```
import java.util.Scanner;
public class demo
{
```

```
public static void main(String args[])
{
System.out.println("Enter two numbers:  ");
Scanner sc = new Scanner(System.in);
int n1 = sc.nextInt();
int n2 = sc.nextInt();
int lcm=0;
int max = Math.max(n1 , n2);
while(max<=n1*n2){
if (max%n1==0 && max%n2==0){
 lcm = max;
    break;
    }
max++;
   }
System.out.println("LCM of "+ n1 + " and " + n2+ " is " + lcm );
  }
}
```

8. HCF

```
import java.util.Scanner;
public class demo
{
public static void main(String args[])
{
System.out.println("Enter two numbers:  ");
Scanner sc = new Scanner(System.in);
int n1 = sc.nextInt();
int n2 = sc.nextInt();
int hcf=1;
int min = Math.min(n1,n2);
for(int i = 1; i<= min ; i++){
    if(n1%i==0 && n2%i == 0){
        hcf  =i;
    }
   }
System.out.println(" HCF of the numbers is :" + hcf);
  }
  }
```

9. Vowel or consonant

```
import java.util.Scanner;
public class demo
{
public static void main(String args[])
{
System.out.println("Enter a letter :  ");
Scanner sc = new Scanner(System.in);
char c = sc.next().trim().charAt(0);
sc.close();
if(c=='a' || c=='e' || c=='i' || c=='o'|| c=='u'){
    System.out.println(c + " is a vowel !");
  }
else {
    System.out.println( c + " is a consonant !");
  }
 }
}
```

10. Find factors of a number.

```
import java.util.Scanner;
public class demo
{
public static void main(String args[])
{
System.out.print("Enter the number to check if it is perfect or not  :  ");
Scanner sc = new Scanner(System.in);
int num  =sc.nextInt();
sc.close();
int sum = 0;
int factor =0;
for (int i=1; i<= Math.sqrt(num); i++){
    if(num%i==0){
    factor = num/i;
    System.out.println(i);
    System.out.println(factor);
   }
  }
 }
}
```

11. Check if a number if perfect or not.

```
import java.util.Scanner;
public class demo
{
public static void main(String args[])
{
System.out.print("Enter the number to check if it is perfect or not  :  ");
Scanner sc = new Scanner(System.in);
int num  =sc.nextInt();
sc.close();
int sum = 0;
int factor =0;
//System.out.println(" factors are : ");
for (int i=2; i<= Math.sqrt(num); i++){
    if(num%i==0){
    factor = num/i;
    sum=sum+i+ factor;
   // System.out.println(i);
   // System.out.println(factor);
   }
  }
sum = sum+1;
// System.out.println("sum is :" + sum);
if (num==sum){
    System.out.println(num + " is a perfct number !");
}
else{
   System.out.println(num + " is not a perfect  number !");
   }
 }
}
```

12. Check if a year is leap or not.

```
import java.util.Scanner;
public class demo
{
public static void main(String args[])
{
```

```
System.out.print("Enter the year to check if it is leap or not  :  ");
Scanner sc = new Scanner(System.in);
int year  =sc.nextInt();
sc.close();
if(year%400==0){
    System.out.println(year + " is a leap year.");
}
else if(year%100==0) {
  System.out.println(year + " is not a leap year");
}
else if (year%4==0){
    System.out.println(year + " is a leap year.");
}
else{
    System.out.println(year  + " is not a leap year.");
 }
 }
}
```

13. Write a program to print the sum of negative numbers, sum of positive even numbers and the sum of positive odd numbers from a list of numbers (N) entered by the user. The list terminates when the user enters a zero.

```
import java.util.Scanner;
public class demo
{
public static void main(String args[])
{
int sume = 0;    //sum of even numbers
int sumo = 0;    //sum of odd numbers
int sumn = 0;   //sum of negative numbers
Scanner sc = new Scanner(System.in);
while(true){
    System.out.print("Enter the number :  ");
    int n = sc.nextInt();
    if (n%2==0 && n>0){
        sume = sume+n;
    }
    else if (n%2!=0 && n>0){
        sumo = sumo+n;
    }
    else if (n<0){
        sumn = sumn+n;
    }
    if(n==0){
        break;
       }
  }
  sc.close();
System.out.println(" Sum of even numbers is : " + sume);
System.out.println(" Sum of odd numbers is : " + sumo);
System.out.println(" Sum of negative numbers is : " + sumn);
 }
}
```

FUNCTIONS -

04-functions.md

1. Define two methods to print the maximum and the minimum number respectively among three numbers entered by the user.

```
import java.util.Scanner;
```

```
public class functions {
    public static void main(String[] args) {
        System.out.println(" Enter the numbes you want to compare : ");
        Scanner sc = new Scanner(System.in);
        int a  = sc.nextInt();
        int b  = sc.nextInt();
        int c = sc.nextInt();
        int min_ans = min(a,b,c);
        int max_ans = max(a,b,c);
        sc.close() ;
        System.out.println("minimum of "+ a + ","+ b+ ","+ c+ " is "+ min_ans);
        System.out.println("maximum of "+ a + ","+ b+ ","+ c+ " is "+ max_ans);
    }

    public static int min(int d, int e , int f) {
        int mini = 0;
       if (d<e && d<f){
        mini = d;;
       }
       if (e<d && e<f){
       mini =  e ;
       }
       if (f<d && f<e){
         mini = f ;
       }
    return mini ;
        }

    public static int max(int d, int e , int f) {
        int maxi = 0;
        if (d>e && d>f){
            maxi = d;
           }
        if (e>d && e>f){
           maxi=  e ;
           }
        if (f>d && f>e){
             maxi = f ;
           }
        return maxi ;
            }
}
```

2. Define a program to find out whether a given number is even or odd.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the number you want to check : ");
        Scanner sc = new Scanner(System.in);
        int a  = sc.nextInt();
        eveodd(a);
        sc.close() ;


}
 public static void eveodd( int n) {
    if(n%2== 0 ) {
      System.out.println(n + " is even !");
    }
    else{
        System.out.println(n + " is odd !");
    }
 }
}
```

3. A person is eligible to vote if his/her age is greater than or equal to 18. Define a method to find out if he/she is eligible to vote.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter your age : ");
        Scanner sc = new Scanner(System.in);
        int age  = sc.nextInt();
        boolean b  = eligibility (age) ;
        sc.close() ;
        System.out.println(" Are you eligible to vote ? : " + b);

}
 public static boolean eligibility(int n) {
    return n>=18 ;
 }
}
```

4. Write a program to print the sum of two numbers entered by user by defining your own method.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the two numbers: ");
        Scanner sc = new Scanner(System.in);
        int a  = sc.nextInt();
        int b = sc.nextInt() ;
        int ans = add(a,b) ;
        sc.close() ;
        System.out.println("addition of the two numbers is "+ ans);

}
 public static int add (int d, int f) {
    int e  =d+ f;
    return e ;
 }
}
```

5. Define a method that returns the product of two numbers entered by user.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the two numbers: ");
        Scanner sc = new Scanner(System.in);
        int a  = sc.nextInt();
        int b = sc.nextInt() ;
        int ans = add(a,b) ;
        sc.close() ;
        System.out.println(ans );

}
 public static int add (int d, int f) {
    int e  =d*f;
    return e ;
 }
```

6. Write a program to print the circumference and area of a circle of radius entered by user by defining your own method.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the radius : ");
        Scanner sc = new Scanner(System.in);
        int r = sc.nextInt();
        sc.close() ;
        double pi  = 3.14159265359;
        double area_c =  area(r, pi);
        double circum_c = circumference(r,pi) ;
        System.out.println(" the area of the circle with radius "+ r + " is "+ area_c);
        System.out.println(" the circumference of the circle with radius "+ r + " is "+ circum_c);

}
 public static double area(int n , double pi ) {
    double a   = pi*n*n ;
    return a ;
 }
 public static double circumference(int n , double pi){
    double circum = 2*pi*n ;
    return circum ;
 }
}
```

7. Define a method to find out if a number is prime or not.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the number : ");
        Scanner sc = new Scanner(System.in);
        int num  = sc.nextInt();
        boolean b  = prime(num);
        System.out.println(" Is " + num + " prime ? : " + b);
        sc.close();
 }
 public static boolean prime (int n ){
    int flag = 0;

        if (n==2){
             flag = 1;
            }
        for ( int i=3 ; i<=Math.sqrt(n);i++){
        if (n%i==0){
            flag  = 0;
            break ;
        }
        else{
            flag  =1 ;
        }
    }
    return (flag == 1) ;
 }
}
```

8. Write a program that will ask the user to enter his/her marks (out of 100). Define a method that will display grades according to the marks entered as below:

Marks	Grade
91-100	AA
81-90	AB
71-80	BB
61-70	BC
51-60	CD
41-50	DD
<=40	Fail

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter your marks out of 100 : ");
        Scanner sc = new Scanner(System.in);
        int num  = sc.nextInt();
        grades(num );
        sc.close();
}
public static void grades(int n ){
    System.out.print(" your grade is : ");
    if (n<=100 && n>=91){
        System.out.println("AA");
    }
    else  if (n<=90 && n>=81){
        System.out.println("AB");
    }
    else if (n<=80 && n>=71){
        System.out.println("BB");
    }
    else  if (n<=70 && n>=61){
        System.out.println("BC");
    }
    else  if (n<=60 && n>=51){
        System.out.println("CD");
    }
    else if (n<=50 && n>=41){
        System.out.println("DD");
    }
    else{
        System.out.println(" Fail ");
    }
}

}
```

10. Write a program to print the factorial of a number by defining a method named 'Factorial'.
    Factorial of any number n is represented by n! and is equal to 1 * 2 * 3 * .... * (n-1) *n.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the number : ");
        Scanner sc = new Scanner(System.in);
        int num  = sc.nextInt();
        int fac = Factorial(num );
        sc.close();
        System.out.print(" the factorial of " + num + " is : " + fac);
}
public static int Factorial (int n ){

    int prod =1 ;
    if (n==0){
```

```
        return 1;
    }
    for ( int i=n; i>0; i --){
        prod  = prod*i ;
    }
    return  prod ;
  }

}
```

11. Write a function to find if a number is a palindrome or not. Take number as parameter.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the number : ");
        Scanner sc = new Scanner(System.in);
        int num  = sc.nextInt();
        boolean b = palindrome(num );
        sc.close();
        System.out.print(" Is " + num + " a palindrome? :"+ b);
}

public static boolean palindrome (int n ){
    String s = Integer.toString(n);
   int i = 0;
   int flag = 0;
   int j  = s.length()-1;
   while(i<j){
   if (s.charAt(i)==s.charAt(j)){
    flag = 0 ;
      }
   else {
    flag  =1 ;
    break;
      }
   i++;
   j--;
      }
   return flag == 0;
   }
}
```

12. Write a function to check if a given triplet is a Pythagorean triplet or not.

```
import java.util.Scanner;

public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the sides of the triangle: ");
        Scanner sc = new Scanner(System.in);
        int a  = sc.nextInt();
        int b  = sc.nextInt();
        int c = sc.nextInt() ;
        int max = 0;
        sc.close() ;
        if (a>b && a>c){
            max = a ;
            a = 0;
        }
        else if (b>c && b>a){
            max = b;b = 0;
        }
        else if (c>a && c>b){
            max = c ;
            c = 0;
```

```
        }
        //System.out.println(a+"," + b+  "," +c+ ","+ max);


    boolean boo  = Pytrip( a ,  b,  c, max ) ;
    System.out.println(" are the sides Pythagorean triplets ?: "+ boo);
    }

    public static boolean Pytrip( int d, int e , int f, int g){
        int sq = d*d + e*e + f* f;
        return (g*g == sq);
    }
}
```

13. Write a function that returns all prime numbers between two given numbers.

```
import java.util.Scanner;


public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter the limit : ");
        Scanner sc = new Scanner(System.in);
        int a  = sc.nextInt();
        int b  = sc.nextInt();
        primes(a,b);
        sc.close();
    }
    public static void primes( int d , int f){

        for (int i = d+1 ; i< f && i>d ; i++){
            if(i==1 || i==0)
            { continue ; }
            int flag=0;
            for (int j=2 ; j<=Math.sqrt(i);j++){
              if(i%j==0){
              flag = 1;
              break ;
          }
         }
          if (flag ==0){
            System.out.println(i+ " is prime");
        }
    }
  }
}
```

14. Write a function that returns the sum of first n natural numbers.

```
import java.util.Scanner;


public class functions {
    public static void main(String[] args) {
        System.out.print(" Enter n : ");
        Scanner sc = new Scanner(System.in);
        int num  = sc.nextInt();
        int sum = add(num);
        System.out.println(" The addition of first "+ num + " natural numbers is "+ sum);
        sc.close();
    }
    public static int add( int n ){
        int sum = 0;
        for (int i=1;i<=n;i++){
            sum=sum+i;
        }
        return sum;
```

```
    }
  }
```

NOTES -

Array -

swapping two numbers (the first and the last here)

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Arrays;

public class arraylist {
    public static void main(String[] args) {
       int arr[]= {3,5,4,67,21} ;
       swap(arr , 0, 4);
    }

    static void swap(int[] arr, int ind1 , int ind2){
      int temp = arr[ind1];
      arr[ind1]= arr[ind2];
      arr[ind2]= temp;
      System.out.println(Arrays.toString(arr));
    }

}
```

print the maximum number in the array -

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Arrays;

public class arraylist {
    public static void main(String[] args) {
       int arr[]= new int[5] ;
    Scanner sc  = new Scanner(System.in);
    System.out.println(" enter the numbers: ");
       for(int i =0;i<5;i++){
        arr[i] = sc.nextInt();
       }
       sc.close();
       max(arr);
    }

    static void max(int[] arr){
        int greatest = arr[0];
       for(int i =0;i< arr.length;i++){
        if( arr[i] > greatest){
            greatest = arr[i];
        }

    }
        System.out.println(" maximum number is :" + greatest);
   }
}
```

Reverse numbers in an array -

```
import java.util.Scanner;
import java.util.Arrays;

public class arraylist {
    public static void main(String[] args) {
       int arr[]= new int[5] ;
    Scanner sc  = new Scanner(System.in);
    System.out.println(" enter the numbers: ");
       for(int i =0;i<5;i++){
        arr[i] = sc.nextInt();
       }
       sc.close();
       reverse(arr);
    }
    static void reverse(int[] arr){
        int i =0;
        int j = arr.length-1;
        while(i<j){
            int temp = arr[i];
            arr[i]= arr[j];
            arr[j] = temp;
            i++;
            j--;
        }
        System.out.println(Arrays.toString(arr));
    }
}
```

arraylist -

can be used when we do not know how many items to add.

similar to vectors in c++.

```
import java.util.ArrayList;
import java.util.Scanner;

public class arraylist {
    public static void main(String[] args) {
       // Scanner sc = new Scanner(System.in);
        // Syntax
        ArrayList<Integer> list = new ArrayList<>(5);

       list.add(67);
       list.add(234);
       list.add(654);
       list.add(43);
       list.add(654);
       list.add(8765);

       System.out.println(list.contains(765432));
       System.out.println(list);
       list.set(0, 99);

       list.remove(2);

       System.out.println(list);

    }
}
```

taking input and print numbers -

```
import java.util.ArrayList;
import java.util.Scanner;

public class arraylist {
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        // Syntax
        ArrayList<Integer> list = new ArrayList<>(5);
        // we can add as number of items as we want
System.out.println(" add numbers to your list: ");
      for (int i=0; i< 5; i++){
        list.add(sc.nextInt());
      }
System.out.println("print the numbers:");
      for (int i=0; i< 5; i++){
       System.out.println(list.get(i));
      }

    }
}
```

Multi-dimentaional arraylist -

```
import java.util.ArrayList;
import java.util.Scanner;

public class arraylist {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //multidimentional arraylist
        ArrayList<ArrayList<Integer>> list = new ArrayList<>();

        //initialization of array lists inside the arraylist

        for (int i=0;i<3;i++){
            list.add(new ArrayList<>());
        }

       // add elements
       System.out.println(" enter numbers: ");
       for (int i=0;  i<3;i++){
        for (int j=0;j<4;j++){
            list.get(i).add(sc.nextInt());
        }
       }
       System.out.println(list);
    sc.close();
    }
}
```

output -

```
PS C:\DSA> javac arraylist.java
PS C:\DSA> java arraylist
 enter numbers:
32
456
12
78
45
743
89
866
112
45
906
674
[[32, 456, 12, 78], [45, 743, 89, 866], [112, 45, 906, 674]]
```

ASSIGNMENT-

04- arrays.md (leetcode)

1920. Build Array from Permutation

```
class Solution {
    public static void main(String[] args) {
    int nums[]= {1,2,0,5,3,4};
       buildArray(nums);
    }

    public static int[] buildArray(int[] nums){
        int ans[]= new int[nums.length];
        for (int i =0; i<nums.length;i++){
            ans[i]= nums[nums[i]];
        }
        return ans;
}
}
```

1929. Concatenation of Array

```
class Solution {
    public int[] getConcatenation(int[] nums) {

        int n = nums.length;
        int[] ans = new int[2*n] ;
    for(int i=0;i< n ; i++){
       ans[i]=nums[i] ;
    }
    int i = 0;
    while(i<n){
        ans[i+n]  =nums[i];
        i++;
    }
    return ans;
    }
}
```

1480. Running Sum of 1d Array

```
class Solution {
    public int[] runningSum(int[] nums) {
        int n = nums.length;
        int[] runningSum = new int[n];
        runningSum[0] = nums[0];
        for (int i = 1; i<n;i++ )  {
         runningSum[i] = nums[i]+runningSum[i-1];
        }
        return runningSum;
   }
  }
```

100% faster solution -

```
//not my solution
class Solution {
    public int[] runningSum(int[] nums) {
        for (int i = 1; i < nums.length; i++) {
            nums[i] += nums[i - 1];
        }
        return nums;
    }
}
```

1672. Richest Customer Wealth

```
class Solution {
    public int maximumWealth(int[][] accounts) {
         int m = accounts.length;
    int n = accounts[0].length;
         int max = 0;

     for(int i =m-1 ;i>=0;i--){
        int sum = accounts[i][0];
        for (int j =n-1 ;j>0;j--){
       sum = accounts[i][j]+sum;
            }
            System.out.println(sum);
            if(sum>max){
                max=sum;
        }
     }
       return max;
    }
}
```

or

```
// not my solution
class Solution {
    public int maximumWealth(int[][] accounts) {
        int res = 0;
        for(int i =0;i<accounts.length;i++){
            int temp = 0;
            for(int j = 0;j<accounts[i].length;j++){
                temp+=accounts[i][j];
            }
            res = Math.max(res,temp);
        }
        return res;
    }
}
```

1431. Kids With the Greatest Number of Candies

```
class Solution {
    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {
    ArrayList<Boolean> list = new ArrayList<Boolean>(candies.length)  ;
    int max = 0;
    for(int i =0;i<candies.length;i++){
     if(candies[i]>=max){
        max = candies[i];
     }
```

```
    }
    Boolean[] result = new Boolean[candies.length];
    for (int i =0; i< candies.length;i++){
        if (candies[i]+ extraCandies >= max){
         result[i]=true ;
        }
        else{
            result[i]= false;
        }
    }
        for (int i =0;i<result.length; i++){
            list.add(result[i]);
        }
        return list;
    }
}
```

1470. Shuffle the Array

```
class Solution {
    public int[] shuffle(int[] nums, int n) {
        int[] arr=new int[2*n];
        int i=0;
        int j=n;
        for(int k=0;k<2*n;k=k+2){
            arr[k]=nums[i];
            arr[k+1]=nums[j];
            i++;
            j++;
        }
        return arr;
    }
}
```

1512. Number of Good Pairs

```
class Solution {
    public int numIdenticalPairs(int[] nums) {
    int count = 0;
    for(int i =0;i<nums.length;i++){
        for (int j = 1; j< nums.length ; j++){
           if(i<j && nums[i]==nums[j]){
            count++;
           }
        }
    }
  return count;
    }
}
```

1365. How Many Numbers Are Smaller Than the Current Number

```
class Solution {
    public int[] smallerNumbersThanCurrent(int[] nums) {
         int[] arr =  new int[nums.length] ;
```

```
    for(int i =0;i<nums.length;i++){
       int count = 0;
        for (int j = 0; j< nums.length ; j++){
           if(i!=j && nums[i]>nums[j]){
            count++;
           }
           arr[i] = count;
           }
    }
      return arr;
    }
}
```

1389. Create Target Array in the Given Order

```
class Solution {
    public int[] createTargetArray(int[] nums, int[] index) {
int[] target = new int[nums.length];
 ArrayList<Integer> list = new ArrayList<>();
 for(int i =0;i< nums.length ; i++){
  list.add(index[i],nums[i]);
    }
        for (int i =0;i<nums.length;i++){
            target[i]= list.get(i);
        }
        return target ;
    }
}
```

1832. Check if the Sentence Is Pangram

```
class Solution {
    public boolean checkIfPangram(String sentence) {
   int flag = 0;
  if(sentence.length()<26){
    return false;
  }
  for( char j = 'a' ; j<='z'; j++){
    flag=0;
    for (int i = 0; i<sentence.length() ; i++){
      if(j==sentence.charAt(i)){
        flag= 1;
        break;
      }
    }
    if(flag==0){ break;}
  }
 return(flag==1);
}
}
```

1773. Count Items Matching a Rule

```
class Solution {
    public int countMatches(List<List<String>> items, String ruleKey, String ruleValue) {
```

```
         int count = 0;

        for (List<String> item : items){
            if (ruleKey.equals("type") && item.get(0).equals(ruleValue)) count++;
            if (ruleKey.equals("color") && item.get(1).equals(ruleValue)) count++;
            if (ruleKey.equals("name") && item.get(2).equals(ruleValue)) count++;
        }

        return count;
    }
}
```

1732. Find the Highest Altitude

```
class Solution {
    public int largestAltitude(int[] gain) {
      int[] arr  = new int[gain.length+1];
      arr[0]= 0;
      int sum = 0;
      for (int i =0;i<gain.length;i++){
          sum = sum+gain[i];
          arr[i+1] =sum;
      }
      int max = 0;
      for (int i =0;i<arr.length; i++){
        if (arr[i]>max){
          max= arr[i];
        }
      }
      return max;
    }
}
```

832. Flipping an Image

```
class Solution {
    public int[][] flipAndInvertImage(int[][] image) {
int  n = image.length;
  int [][] arr = new int[n][n];
  // for horizontal reverse of the array
    for (int i =0;i<n;i++){
      int m =n-1;
      for ( int j =0;j<n;j++){
        arr[i][j] = image[i][m];
        m-=1;
          }
}
//System.out.println(Arrays.deepToString(arr));

// to swap 1 and 0
for (int i  =0;i<n;i++){
  for (int j=0;j<n;j++){
    if(arr[i][j]==0){
      arr[i][j]=1;
    }
    else if( arr[i][j]==1){
      arr[i][j]=0;
    }
  }
}
        return arr;
    }
}
```

1572. Matrix Diagonal Sum

```
class Solution {
    public int diagonalSum(int[][] mat) {
  int  n = mat.length;
  int sum  =0;
   if (n==1){sum = mat[0][0];}
   else {
       // 0,0+1,1+2,2+...+k,k
   for ( int i =0; i<n;i++){
    for ( int j =0;j<n;j++){
       if(i==j){
       sum += mat[i][j];
       }
    }
   }
  int i =0;
  int k =n-1;
       // for 5x5 matrix - 0,4+1,3+2,2+3,1+40
  while(k>=0 && i < n ){
    sum+=mat[i][k];
    i++;
    k--;
  }
  if (n%2!=0 && n!=1){ // for an odd matrix, centre is repeated twice
    sum -= mat[n-(n/2)-1][n-(n/2)-1];   // 5x5 matrix centre = 2,2
  }
}
return sum;

    }
}
```

1295. Find Numbers with Even Number of Digits

```
class Solution {
    public int findNumbers(int[] nums) {
         int  n = nums.length;
  int [] countarr = new int[n];
   for (int i = 0; i< n;i++){
    int count=0;
    int digit = nums[i];
      while(digit>0){
        int rem = digit%10;
        count++;
        digit/=10;
      }
      countarr[i]= count;
   }
   System.out.println(Arrays.toString(countarr));
int even  =0;
for (int i =0; i<n;i++){
 if (countarr[i]%2==0){
  even++;
 }
}
return even;
    }
}
```

867. Transpose Matrix

```
class Solution {
    public int[][] transpose(int[][] matrix) {
        int[][] res = new int[matrix[0].length][matrix.length];
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                res[j][i] = matrix[i][j];
            }
        }
        return res;
    }
}
```

989. Add to Array-Form of Integer

```
class Solution {
    public List<Integer> addToArrayForm(int[] num, int k) {

    int n = num.length;
    int i = n-1;
    List<Integer> sol = new ArrayList<>();
    while(i >= 0 || k > 0) {
        if(i >= 0) {
        sol.add((num[i] + k) % 10);
        k = (num[i] + k) / 10;
    } else {
        sol.add(k % 10);
        k = k / 10;
        }
        i--;
    }
    Collections.reverse(sol);
    return sol;

    }
}
```

1252. Cells with Odd Values in a Matrix

```
class Solution {
    public int oddCells(int m, int n, int[][] indices) {
        int[][] matrix = new int[m][n];
        for(int i=0; i<indices.length; i++){
            row(matrix, indices[i][0]);
            col(matrix, indices[i][1]);
        }
        int odd = 0;
        for(int i=0; i<m; i++){
            for(int j=0; j<n; j++){
                if(matrix[i][j]%2==1)
                    odd++;
            }
        }
        return odd;
    }

    void row(int[][] mat, int rowNo){
        for(int i=0; i<mat[rowNo].length; i++){    //j<no of  columns
```

```
            mat[rowNo][i]++;
        }
    }

    void col(int[][] mat, int colNo){
        for(int i=0; i<mat.length; i++){       // i< no of rows
            mat[i][colNo]++;
        }
    }
}
```

1886. Determine Whether Matrix Can Be Obtained By Rotation

```
class Solution {
    public boolean findRotation(int[][] mat, int[][] target) {
         for(int i=0; i<4; i++){
    if(isEqual(mat,target)) return true;
   mat =  rotate(mat);
    }
    return false;
}


public static int[][] rotate(int[][] matrix){
    // transpose
int n  =matrix.length ;
int[][] result  = new int[n][n];

for ( int i = 0; i< n; i++){
    for ( int j =0; j< n;j++) {
        result[i][j] = matrix[j][i];
    }
}
// swap columns
int[][] swap = new int[n][n];
int m = n;
for ( int i  =0; i<= m-m/2-1; i++){  // no of column swappings according to the matrix
    int j =0;
    n--;
    while(j<m&&n>=0){
    int temp = result[j][i];
    swap[j][i]= result[j][n];
    swap[j][n] =temp;
    j++;
    }
}
    return swap;

// System.out.println("90 degrees rotation - ");
// System.out.println(Arrays.deepToString(swap));;
}

//     Function to check whether two matrix are equal
public static boolean isEqual(int[][] m1 , int[][] m2){
    if(m1.length != m2.length) return false;
    if(m1[0].length != m2[0].length) return false;
    for(int row=0; row<m1.length; row++){
        for(int col = 0; col<m1[0].length; col++){
            if(m1[row][col] != m2[row][col]) return false;
        }
    }
    return true;
    }
}
```

1. Two Sum

```
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int[] indices = new int[2];
        for( int i =0;i<nums.length ;i++){
            for (int j = i+1; j< nums.length ;j++ ){
                if(nums[i]+nums[j] == target){
                    indices[0]=i;
                    indices[1] = j;
                }
            }
        }
        return indices;
    }
}
```

1304. Find N Unique Integers Sum up to Zero

```
class Solution {
    public int[] sumZero(int n) {
        int[] arr = new int[n];
        int start = 0;
        int end = n - 1;

        while(start < end){
            arr[start] = start + 1;
            arr[end] = arr[start] * (-1);
            start++;
            end--;
        }
        return arr;
    }
}
```

1380. Lucky Numbers in a Matrix

```
//PARTIALLY WRONG CODE THAT I AM PROUD OF ; *HOPE SOMEDAY I WILL BE ABLE TO FIX IT*

import java.util.ArrayList;
import java.util.Scanner;
import java.util.Arrays;
import java.util.Collections;

public class arraylist {
public static void main(String[] args) {
    //int[][] matrix = {{3,6},{7,1},{5,2},{4,8}};
int[][] matrix = {{1,10,4,2},{9,3,8,7},{15,16,17,12}};
int n = matrix.length;
// row check

int index=0;
int[] arr  = new int[matrix.length];
int indices[] = new int[matrix.length] ;
for ( int i  =0; i< n; i++ ){
    int  min=matrix[i][0] ;
    index = 0;
  for ( int j =1; j< matrix[0].length; j++){
      if(min>matrix[i][j]){

        min  = matrix[i][j];
        index = j;
```

```
      }
  }
    arr[i] = min;
    indices[i]= index;

}
System.out.println(Arrays.toString(arr));
System.out.println(Arrays.toString(indices));

ArrayList<Integer> list  =new ArrayList<>(1);

int max=0;
for( int i =0; i< n; i++){
    max  = arr[i];
   for ( int j = 0; j< arr.length; j++){
        if(matrix[j][indices[i]]>max ){
            flag =1;
            break;
        }
    }
}

}
}
```

correct solution -

```
class Solution {
    public List<Integer> luckyNumbers (int[][] matrix) {
     List<Integer> result = new ArrayList<Integer>();
//rows[] and col[] arrays are used to save row_min and col_max values
int rows[] = new int[matrix.length];
int columns[] = new int[matrix[0].length];

//to find row_min values and adding into row[]
for (int i = 0; i < matrix.length; i++) {
    int min = 999999999;
    for (int j = 0; j < matrix[i].length; j++) {
         min = Math.min(matrix[i][j], min);
    }
    rows[i] = min;
}

//To find col_max values and adding into col[]
 for(int i=0; i < matrix[0].length;i++){
     int max = 0;
    for (int j = 0; j < matrix.length; j++) {
         max = Math.max(matrix[j][i], max);
    }
    columns[i] = max;
}

//to store result value because return type is arraylist

 //To compare arrays(row[] and col[]
for(int i=0;i<matrix.length;i++){
    for(int j=0; j < matrix[0].length;j++){
        if(rows[i]==columns[j]){
          result.add(rows[i]);
          }
    }
}
return result;
}
}
```

53. Maximum Subarray

```
class Solution {
    public int maxSubArray(int[] nums) {
 int max = Integer.MIN_VALUE;
        int currsum = 0;

        for(int i=0;i<nums.length ;i++){
            currsum += nums[i];
            max = Math.max(currsum,max);

            if(currsum<0) {
            currsum = 0;
            }
        }
        return max;
}
}
```

566. Reshape the Matrix

```
class Solution {
    public int[][] matrixReshape(int[][] mat, int r, int c) {
  int[][] reshape  = new int[r][c];

  ArrayList<Integer> list = new ArrayList<>();
  for( int  i =0; i< mat.length; i++){
    for( int j =0; j< mat[i].length; j++){
        list.add(mat[i][j]);
    }
  }

  if( list.size() == r*c){
    int index  =0;
    while(index<r*c)
for ( int i =0; i<r;i++){
    for ( int j =0; j< c; j++){
{
        reshape[i][j]= list.get(index);
        index++;
    }
}
 }
  }

else {
   reshape =mat;
}
return reshape;
   }
}
```

66. Plus One

```
class Solution {
    public int[] plusOne(int[] digits) {
for (int i = digits.length - 1; i >= 0; i--) {
  if (digits[i] < 9) {   // from the last , checking if digit at n-1 less than 9. if not adding 1 to the end and returning the array.
    digits[i]++;
    return digits;
  }
  digits[i] = 0; // if array[i] = 9, then changing that element to 0 and going back to the 'for loop' for incrementing the previous element
}
```

```
digits = new int[digits.length + 1]; // if all digits of the array are 9 , increasing the loop size by 1 and adding 1 to the first place of
digits[0] = 1;
return digits;

    }
}
```

26. Remove Duplicates from Sorted Array

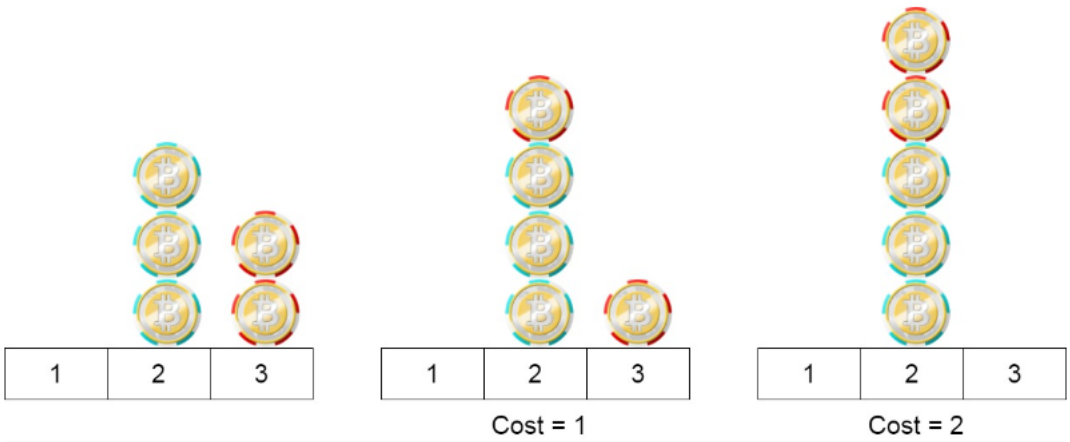
```
class Solution {
    public int removeDuplicates(int[] nums) {
        int n = nums.length;
    int unique = 1;
        for(int i = 0 ; i < n-1 ; i++ ){
            if(nums[i] != nums[i + 1]){         //When we get unique No.
                nums[unique] = nums[i+1];          //update previous pointer with new element
                unique++;
            }
        }
        return unique ;
    }
}
```

1217. Minimum Cost to Move Chips to The Same Position

cost for 1 shift =1

cost for 2 shifts = 0

so stack shifting even position to even and odd position to add will cost 0 . the only cost is for shifting coins from odd to even or even to odd. So , if odd position has 5 coins after complete stacking of coins at odd positions and even positions has 3 coins , we will stack the even coins at odd position one by one. So , minimum of odd coins and even coins is cost.



Input: position = [2,2,2,3,3]
Output: 2
Explanation: We can move the two chips at position 3 to position 2. Each move has cost = 1. The total cost = 2.

```
class Solution {
    public int minCostToMoveChips(int[] position) {
           int even=0, odd=0;
      for ( int i =0; i< position.length; i++){

          if(position[i]%2==0){
              even++;
          }
          else odd++;
      }
        int cost = Math.min(even, odd);
        return cost;

    }
}
```

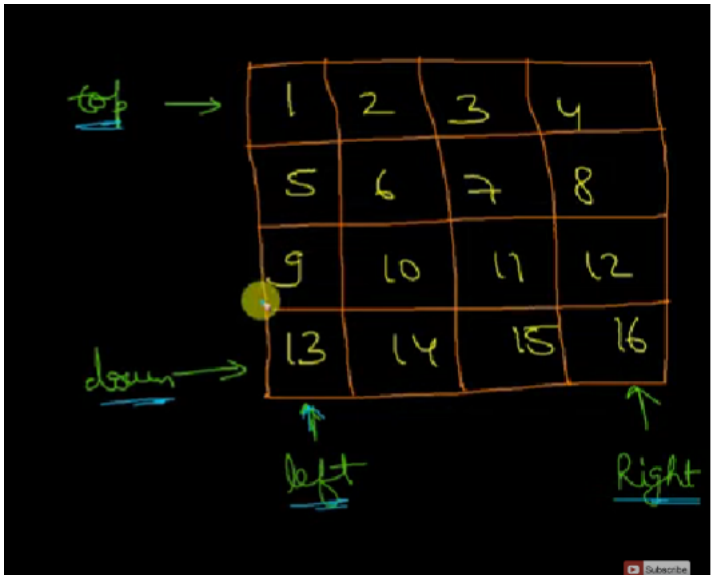
1854. Maximum Population Year

```
class Solution {
    public int maximumPopulation(int[][] logs) {
          int res[] = new int[101];

        for(int[] log : logs){   // loops thru each element of the 2d array
            res[log[0]-1950]++;   // index of birth year (0 th col) is incremented by 1
            res[log[1]-1950]--;   //index of death year (1th col) in decremented by 1.
        }

        int max = res[0];
        int year = 1950;

        for(int i=1; i<101; i++){
            res[i] += res[i-1];
            if(res[i] > max){
                    max = res[i];
                    year = i+1950;
            }
        }
        return year;
    }
}
```

INTERMEDIATE -

54. Spiral Matrix

```
direction indicators  -
- dir = 0 ( towards right)
- dir =1( towards south)
- dir  =2 ( towards left)
- dir  =3 ( towards north)
```



youtube reference -

https://youtu.be/1ZGJzvkcLsA

```
class Solution {
    public List<Integer> spiralOrder(int[][] matrix) {
       int m  =matrix.length;
int n = matrix[0].length;

ArrayList<Integer> list  = new ArrayList<>();
int top = 0, down = m-1 , left  = 0, right  = n-1 ;
int dir=0;


while(top<=down && left<=right){
if(dir==0){
for( int i  = left ; i<=right ;i++){
list.add(matrix[top][i]);
}
top++;
}

else if(dir==1){
    for(int i = top; i<=down; i++){
   list.add(matrix[i][right]);
    }
right--;
}


else if( dir == 2){
    for ( int i = right; i>= left ; i--){
        list.add(matrix[down][i]);
    }
    down--;
}


else if(dir==3){
    for( int i = down; i>= top; i--){
        list.add(matrix[i][left]);
    }
    left ++;
}
```

```
dir=(dir+1)%4;
}
     return list;
    }

}
```

59. Spiral Matrix II

```
class Solution {
    public int[][] generateMatrix(int n) {

int[][] matrix = new int[n][n];

int top = 0, down = n-1 , left  = 0, right  = n-1, dir=0 ;


int j =1;
while(top<=down && left<=right && j<=n*n){
if(dir==0){
for( int i  = left ; i<=right ;i++){
matrix[top][i]=j;
j++;
}
top++;
}

else if(dir==1){
    for(int i = top; i<=down; i++){
   matrix[i][right]=j;
   j++;
    }
right--;
}


else if( dir == 2){
    for ( int i = right; i>= left ; i--){
        matrix[down][i] = j;
        j++;
    }
    down--;
}


else if(dir==3){
    for( int i = down; i>= top; i--){
        matrix[i][left]=j;
        j++;
    }
    left ++;
}
dir=(dir+1)%4;

    }
        return matrix;
}
}
```

885. Spiral Matrix III

```
class Solution {
    public int[][] spiralMatrixIII(int rows, int cols, int rStart, int cStart) {
```

```
 int[][] result= new int[rows*cols][2];
int k = 0;
result[k++] = new int[]{rStart, cStart};


int dir = 0;
/* direction indicators  -
- dir = 0 ( towards right)
- dir =1( towards south)
- dir  =2 ( towards left)
- dir  =3 ( towards north)
*/
int[] direction = {0, 1, 0, -1, 0};    // flow inside the matrix (used for incrementing and decrementing rows or columns) -  direction- ( l
int LorR = 0;                          // incrementing left or right in one direction
while(k < rows * cols){
    if(dir == 0 || dir == 2){
        LorR++;
    }
    for(int i = 0; i < LorR; i++){
        rStart += direction[dir];
        cStart += direction[dir+1];

        if(rStart > -1 && rStart < rows && cStart > -1 && cStart < cols){
            result[k++] = new int[]{rStart, cStart};
        }
    }
    dir = (dir+1) % 4;
}
        return result;
    }
}
```

73. Set Matrix Zeroes

```
class Solution {
 public void setZeroes(int[][] matrix) {
boolean[][] zeromatrix= equalsZero(matrix) ;
for ( int  i =0; i< matrix.length; i++){
    for  ( int j=0; j< matrix[0].length; j++){
        if(zeromatrix[i][j]==true){
            setRowZero( matrix, i);
            setColZero(matrix,j)  ;
        }
    }
    }
    System.out.println(Arrays.deepToString(matrix));
}

public static void  setRowZero(int[][] matrix, int row){ // to set rows 0
    for( int i =0; i< matrix[0].length ;i ++){
        matrix[row][i]=0;
    }
}

public static void setColZero(int[][] matrix, int col){  // to set columns zero
    for( int i =0; i< matrix.length ;i ++){
        matrix[i][col]=0;
    }
}

public static boolean[][] equalsZero(int[][] matrix){   //to set the indices with 0 to true and rest to false
    boolean[][] zeromatrix = new boolean[matrix.length][matrix[0].length];
    for ( int  i =0; i< matrix.length; i++){
    for  ( int j=0; j< matrix[0].length; j++){
        if(matrix[i][j] == 0)
            zeromatrix[i][j]= true;
        else
            zeromatrix[i][j]= false;
    }
    }
```

```
    return zeromatrix;
}
}
```

238. Product of Array Except Self

```
class Solution {
    public int[] productExceptSelf(int[] nums) {
           int n=nums.length;
   int[] result=new int[n];


    int left=1;
    // cumulative multiplication from the left
    for(int i=0;i<nums.length;i++){


            result[i]=left;
            left*=nums[i];
    }


    int right=1;
    //cumulative multipication from the right using the result array to get the required array

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

        result[i]*=right;
        right*=nums[i];
    }
        return result;
    }
}
```

OR (self written code)

```
class Solution {
    public int[] productExceptSelf(int[] nums) {
  int n=nums.length;
   int[] right =new int[n]; // this array will store the product of elements from the right
   int[] result =new int[n];
   int rightmul =1;

   for( int i =n-1; i>=0; i--){
  rightmul = nums[i]*rightmul;
  right[i]= rightmul;
   }

int leftmul=1; // store product of the left elements of index  i

result[0] =right[1];  // first index will always be multiplication of all the elements in its right(right[i])

for( int i=1; i<n;i++){  // can't calculate the last index coz the loop will run for n-2 times (i+1<n constraint).
    leftmul = leftmul*nums[i-1];
    if(i+1<n)
    result[i]=right[i+1]*leftmul;
}
result[n-1] = leftmul  ;    // last element of the array would be the multiplication of all elements at its left
        return result;
    }
}
```

youtube reference -

https://youtu.be/UBkpyXgx0g0

34. Find First and Last Position of Element in Sorted Array

O(n) time -

```
class Solution {
    public int[] searchRange(int[] nums, int target) {
       int[] result =  {-1,-1};

  for( int i =0; i<nums.length;  i++){
    if(target==nums[i]){
        result[0] = i;
        break;
    }

}

    for( int i =nums.length-1 ; i>=0; i--){
        if(target==nums[i]){
            result[1] = i;
            break;
        }

  }
        return result;


    }
}
```

O(logn) - using binary search

```
class Solution {
    public int[] searchRange(int[] nums, int target) {
  int[] result =  {-1,-1};

  int low = 0;
  int high = nums.length-1;

// finding the smallest index

while(low<=high){
int mid  = (low+high)/2;

if(target==nums[mid]){
    result[0]=mid;
    high = mid-1;  // to continue searching in the lower indices
}
else if(target>nums[mid]){
    low =mid+1;
}
else{
    high = mid-1;
}
}
 low = 0;
high = nums.length-1;
```

```
//finding the largest index

while(low<=high){
    int mid  = (low+high)/2;

    if(target==nums[mid]){
        result[1]=mid;
        low = mid+1;       // to continue searching in the higher indices
    }
    else if(target>nums[mid]){
        low =mid+1;
    }
    else{
        high  = mid-1;
    }

  }
        return result;

    }
}
```

55. Jump Game

```
class Solution {
    public boolean canJump(int[] nums) {

 int reachable =0;  // stores the maximum reachable index by an element

        for( int i =0; i< nums.length ;i++){
            if(reachable<i){
                return false;
            }
            reachable  = Math.max(reachable, nums[i]+i); // check which is greater; the previous reachable or the current reachable; curren
        }
       return true;     // this means that reachable!<i , hence returns true.
    }
}
```

youtube reference -

https://youtu.be/muDPTDrpS28

189. Rotate Array

💡 If the prewritten code has void as the return value, you must modify the given argument(in this case array) , and the compiler will automatically check the value of the modified argument.

```
class Solution {
    public void rotate(int[] nums, int k) {
        int  n= nums.length;
        int[] result = new int[n];
```

```
          for( int i =0; i<n;i++){
              result[(i+k)%n] = nums[i];
          }

         for( int i =0; i<n;i++){
             nums[i]= result[i];
         }

    }
}
```

75. Sort Colors

```
class Solution {
    public void sortColors(int[] nums) {
        int n = nums.length;
  int flag = 0;


for( int i =0; i< n-1; i++){          // bubble sort
  for( int j =0; j< n-i-1 ; j++){
    if(nums[j]>=nums[j+1]){
       int temp = nums[j+1];
       nums[j+1]  =nums[j];
       nums[j]= temp;
       flag = 1;
    }
    }
    if (flag==0)
    break;

}

    }
}
```

198. House Robber

```
class Solution {
    public int rob(int[] nums) {

  // dynamic programming

    int[] dp = new int[nums.length+1]; // it is going to store the maximum money robbed in each of its index
    int max =0;
    dp[0] =0;   // if no houses , no money robbed.
    dp[1]= nums[0];  // if only 1 house, max is going to be the money in that house

    for( int i =1 ; i<nums.length;i++){

// comparing the last value of dp to the sum of current value of nums and last to last value of dp(avoiding adding up the money of two cons
        dp[i+1] = Math.max(dp[i], dp[i-1]+ nums[i]) ;
    }
        return(dp[nums.length]);

    }
}
```