ARRAY-Introduction

https://leetcode.com/problems/rotate-array/

```
class Solution {
public static void reverse(int[] nums,int start,int end){
while(end>start){
int temp = nums[start];
nums[start] = nums[end];
nums[end] = temp;
start++;
end--;
}
}
public void rotate(int[] nums, int k) {
k%= nums.length;
reverse(nums,0,nums.length-1);
reverse(nums,0,k-1);
reverse(nums,k,nums.length-1);
}
}
```

https://leetcode.com/problems/removing-minimum-and-maximum-from-array/description/

```
class Solution {


public int minimumDeletions(int[] nums) {


int max_index = 0;


int min_index = 0;


int n = nums.length;
```

```
for (int i = 1; i < n; i++) {

if (nums[i] > nums[max_index])

max_index = i;



if (nums[i] < nums[min_index])

min_index = i;

}

// 3 cases

// - both delete from left

// - both delete from right

// - right and left together



// delete from left side

int rightmost = Math.max(max_index, min_index);

int del_from_left = rightmost + 1;



// delete from right side
```

```
int leftmost = Math.min(max_index, min_index);

int del_from_right = n - leftmost;



// from both sides

int del_from_both = (n - rightmost) + (leftmost + 1);



// return which is the min distance out of 3 cases

return Math.min(

Math.min(del_from_left, del_from_right),

del_from_both);



}

}
```

https://leetcode.com/problems/reverse-string/

```
class Solution {
public void reverseString(char[] s) {
// Iterate from the beginning (i) and end (j) of the array towards the middle.
for (int i = 0, j = s.length - 1; i < s.length / 2; i++, j--) {
// Use a temporary variable (temp) to store the character at index i.
char temp = s[i];

// Replace the character at index i with the character at index j.
s[i] = s[j];
// Replace the character at index j with the character in the temporary variable (temp).
s[j] = temp;
}
// At this point, the character array 's' contains the reversed string.
}
}
```

You are given an integer **T** (number of test cases). You are given array **A** and an integer **B** for each test case. You have to tell whether **B** is present in array **A** or not.**Problem Constraints**

```
import java.io.*;
public class Main {
private static void digitSum() throws IOException
{
int digit,sum;
final BufferedReader br = new BufferedReader( new InputStreamReader(System.in));
final int t = Integer.parseInt(br.readLine());
for(int i=0 ; i<t ;i++)
{
sum = 0;
final int n = Integer.parseInt(br.readLine());
int temp = n;
for( ;temp>0 ; temp=temp/10)
{
digit = temp%10;
sum+=digit;
}
System.out.println(sum);
}
}
public static void main(String[] args) throws IOException{
// TODO Auto-generated method stub
digitSum();
}

}
```

You are given an integer array **A**. You have to find the second largest element/value in the array or report that no such element exists.

import java.util.*;

```
class sec_large{

// Function to print the
// second largest elements
static void print2largest(int arr[],
int arr_size)
{
int i, first, second;

// There should be
// atleast two elements
if (arr_size < 2)
{
System.out.printf(" Invalid Input ");
return;
}

// Sort the array
Arrays.sort(arr);

// Start from second last element
// as the largest element is at last
for (i = arr_size - 2; i >= 0; i--)
{
// If the element is not
// equal to largest element
if (arr[i] != arr[arr_size - 1])
{
System.out.printf("The second largest " +
"element is %d\n", arr[i]);
return;
}
}

System.out.printf("There is no second " +
"largest element\n");
}

// Driver code
public static void main(String[] args)
{
int arr[] = {12, 35, 1, 10, 34, 1};
int n = arr.length;
print2largest(arr, n);
}
}
```

https://leetcode.com/problems/number-of-good-pairs/

```
class Solution {

    public int numIdenticalPairs(int[] nums) {

    int counter = 0;

    for(int i = 0; i < nums.length; i++){

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

            if(nums[i] == nums[j]){

                counter++;

                }

            }

        }

        return counter;

    }
}
```