

PPT Assignment 1

💡 **Q1.** Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Answer

```
import java.util.*;
```

```
public class PPT1 {
```

```
    static int []twosum(int[] nums, int target)
```

```
    {
```

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

```
        {
```

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

```
                if(nums[i]+nums[j]==target)
```

```
                    return new int[]{i,j};
```

```
        }
```

```
        throw new IllegalArgumentException("no pair found");
```

```
    }
```

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

```
int nums[]={2,7,5,9,11};
```

```
int target=9;
```

```
System.out.println(Arrays.toString(twosum (nums,target)));
```

```
}
```

```
}
```

💡 **Q3.** Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with $O(\log n)$ runtime complexity.

Answer

```
public class PPT2 {
```

```
static int Search(int arr[], int target)
```

```
{
```

```
    if(arr.length==0)
```

```
    {
```

```
        return -1;
```

```
    }
```

```
    for(int i=0;i<arr.length;i++)
```

```
    {
```

```
        int element=arr[i];
```

```
        {
```

```
            if(element==target)
```

```

        {
            return i;
        }
    }
}

return -1;
}

public static void main(String[] args) {

    int arr[]={2,3,5,7,8};

    int target=5;
    System.out.println(Search(arr,target)) ;
}

}

```

💡 **Q4.** You are given a large integer represented as an integer array `digits`, where each `digits[i]` is the *i*th digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of digits.

Answer

```

import java.util.*;

public class IncLargeNum {

    static int[] PlusOne(int arr[])

```

```

{
    int n= arr.length;
    for(int i=n-1;i>=0;i--)
    {
        if (arr[i]<9)
        {
            arr[i]++;
        }
        return arr;
    }
    arr[i]=0;
}

int newnum []=new int[n+1];

newnum[0]=1;

return newnum;

}

public static void main(String[] args) {


    int num[]={ 1,9,8,9};

    System.out.println(Arrays.toString(PlusOne(num)));

}

}

```

 **Q5.** You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

Merge nums1 and nums2 into a single array sorted in non-decreasing order.

The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of $m + n$, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n .

Answer

```
static void MergeArray(int nums1[], int nums2[])
```

```
{
```

```
    int m= nums1.length;
```

```
    int n= nums2.length;
```

```
    int P1=m-1, P2=n-1, i=m+n-1;
```

```
    while(P1>=0)
```

```
    {
```

```
        if(P1>=0 && nums1[P1]>nums2[P2])
```

```
        {
```

```
            nums1[i--]=nums1[P1--];
```

```
        }
```

```
    else
```

```
    {
```

```
        nums1[i--]=nums2[P2--];
```

```
    }
```

```
}
```

```
}
```

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

```
}
```

```
}
```

Q6. Given an integer array `nums`, return `true` if any value appears at least twice in the array, and return `false` if every element is distinct.

Answer

```
static boolean Check(int nums[])  
{  
    Arrays.sort(nums);  
    for(int i=0;i<=nums.length-1;i++)  
    {  
        if(nums[i]==nums[i+1])  
            //for(int j=i+1; j<nums.length-1;j++)  
            // if(nums[i]==nums[j])  
                return true;  
    }  
    return false;  
}  
  
public static void main(String[] args) {  
    int nums[]={2,3,4,5,1,2};  
    System.out.println(Check(nums));  
}
```

```
}
```

💡 Given an integer array `nums`, move all 0's to the end of it while maintaining the relative order of the nonzero elements.

Note that you must do this in-place without making a copy of the array.

Answer

```
static void Move(int nums[])
{
    int j=0;

    for(int i=0;i<=nums.length-1;i++)
    {
        if(nums[i]!=0)
        {
            nums[j]=nums[i];
            j++;
        }
    }

    for(; j<nums.length-1;j++)
    {
        nums[j]=0;
    }
}

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

}

}