1. Two Sum

```
class Solution {
   public int[] twoSum(int[] nums, int target) {
      int a[] = 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)
       {
          a[0]=i;
          a[1]=j;
       }
      return a;

   }
}
```

ROMAN TO INTEGER

```
class Solution {
   public int romanToInt(String s) {
      int ans = 0, num = 0;
      for (int i = s.length()-1; i >= 0; i--) {
         switch(s.charAt(i)) {
            case 'I': num = 1; break;
            case 'V': num = 5; break;
            case 'X': num = 10; break;
            case 'L': num = 50; break;
            case 'C': num = 100; break;
            case 'D': num = 500; break;
            case 'M': num = 1000; break;
         }
         if (4 * num < ans) ans -= num;
         else ans += num;
      }
      return ans;

   }
}
```

LONGEST COMMON PREFIX

```
class Solution {
   public String longestCommonPrefix(String[] v) {
      StringBuilder ans = new StringBuilder();
      Arrays.sort(v);
      String first = v[0];
      String last = v[v.length-1];
      for (int i=0; i<Math.min(first.length(), last.length()); i++) {
         if (first.charAt(i) != last.charAt(i)) {
            return ans.toString();
         }
         ans.append(first.charAt(i));
      }
      return ans.toString();
   }
}
```

VALID PARANTHESES

```
class Solution {
   public boolean isValid(String s) {
      Stack<Character> stack = new Stack<Character>();
      for (char c : s.toCharArray()) {
         if (c == '(')
            stack.push(')');
         else if (c == '{')
            stack.push('}');
         else if (c == '[')
            stack.push(']');
         else if (stack.isEmpty() || stack.pop() != c)

            return false;
      }

      return stack.isEmpty();
   }
}
```

REMOVE DUPLICATES FROM SORTED ARRAY
Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]

```
class Solution {
   public int removeDuplicates(int[] nums) {
      int j = 1;
      for (int i = 1; i < nums.length; i++) {
         if (nums[i] != nums[i - 1]) {
            nums[j] = nums[i];
            j++;
         }
      }
      return j;
   }
}
```

REMOVE ELEMENT
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,,]

```
class Solution {
   public int removeElement(int[] nums, int val) {
      int c = 0;
      for (int i = 0; i < nums.length; i++) {
         if (nums[i] != val) {
            nums[c] = nums[i];
            c++;
         }
      }
      return c;
   }
```

```
}
```

FIRST AND LAST ELEMENT POS OF ELE IN SORTED ARRAY

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

MATRIX [ROTATE IMAGE]
123 741
456 852
789 963

```
class Solution {
   public void rotate(int[][] nums) {
      int[][] arr= new int[nums.length][nums.length];
      for(int i=0;i<nums.length;i++){
         int x=nums.length-1-i;
         for(int j=0;j<nums.length;j++){
            arr[j][x]=nums[i][j];
         }
      }
      for(int i=0;i<nums.length;i++){

         for(int j=0;j<nums.length;j++){
            nums[i][j]=arr[i][j];
         }
      }

   }
}
```

MAXIMUM SUBARRAY

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.

Example 3:
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

```
class Solution {
   public int maxSubArray(int[] nums) {
      int sum=0,m=nums[0];
      for(int i=0;i<nums.length;i++)
                        {
                                    sum+=nums[i];
          m=m>sum?m:sum;
         if(sum<0)sum=0;

                        }
      return m;

   }
}
```

PLUS ONE

Input: digits = [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
Incrementing by one gives 123 + 1 = 124.
Thus, the result should be [1,2,4].

9-> [0,1]

```
class Solution {
   public int[] plusOne(int[] digits) {
      for (int i = digits.length - 1; i >= 0; i--) {
            if (digits[i] < 9) {
                        digits[i]++;
                        return digits;
            }
            digits[i] = 0;
}

digits = new int[digits.length + 1];
digits[0] = 1;
return digits;

   }
}
```

BINARY ADD

```
class Solution
{
 public String addBinary(String a, String b)
 {
   StringBuilder sb = new StringBuilder();
   int carry = 0;
   int i = a.length() - 1;
   int j = b.length() - 1;

   while (i >= 0 || j >= 0 || carry == 1)
   {
    if(i >= 0)
      carry += a.charAt(i--) - '0';
```

```
    if(j >= 0)
      carry += b.charAt(j--) - '0';
    sb.append(carry % 2);
    carry /= 2;
   }
   return sb.reverse().toString();
  }
}
```