Arrays

- Concatenation of Array

```
class Solution {
    public int[] getConcatenation(int[] nums) {
        int n=nums.length;
        int[] ans=new int[n*2];
        for(int i=0;i<n*2;i++)
        {
            int x=i%n;
            ans[i]=nums[x];
        }
        return ans;
    }
}
```

- Contains duplicate

```
class Solution {
    public boolean hasDuplicate(int[] nums) {
        Arrays.sort(nums);
        for(int i=0;i<nums.length-1;i++)
        {
            if(nums[i]==nums[i+1])
            {
                return true;
```

```
            }
        }
        return false;
    }
}
```

Or

```
class Solution {
    public boolean hasDuplicate(int[] nums) {
        Set<Integer> s = new HashSet<>();   // Using hashset
        for (int i : nums)
        {
            if(s.add(i)==false) // returns false when duplicate
elements are present
                return true;
        }
        return false;
    }
}
```

- Valid anagram

```
class Solution {
    public boolean isAnagram(String s, String t) {

        if(s.length() != t.length())
            return false;

        HashMap<Character,Integer> map1=new HashMap<>();
        HashMap<Character,Integer> map2=new HashMap<>();
```

```
        for(int i=0;i<s.length();i++)
        {
            char ch=s.charAt(i);
            char ch1=t.charAt(i);
            map1.put(ch,map1.getOrDefault(ch,0)+1);
            map2.put(ch1,map2.getOrDefault(ch1,0)+1);
        }

        for (Character key : map1.keySet())
        {
            if (!map1.get(key).equals(map2.get(key)))
                return false;
        }
        return true;
    }
}
```

Or

```
class Solution {
    public boolean isAnagram(String s, String t) {
        if(s.length() != t.length())
            return false;

        HashMap<Character,Integer> map=new HashMap<>();

        for(int i=0;i<s.length();i++)
        {
            char ch=s.charAt(i);
```

```
            map.put(ch,map.getOrDefault(ch,0)+1);
        }

        for(char c : t.toCharArray())
        {
            if(map.containsKey(c)==false)
                return false;
            map.put(c,map.get(c)-1);
            if(map.get(c)==0)
                map.remove(c);
        }
        return map.isEmpty();
    }
}
```

Or

```
class Solution {
    public boolean isAnagram(String s, String t) {
        if(s.length() != t.length())
            return false;

        char[] ss=s.toCharArray();
        char[] tt=t.toCharArray();
        Arrays.sort(ss);
        Arrays.sort(tt);
        return Arrays.equals(ss,tt);
    }
}
```

- Two sum

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {

        HashMap<Integer,Integer> map=new HashMap<>();
        int[] arr=new int[2];

        for(int i=0;i<nums.length;i++)
        {
            int x=target-nums[i];
            if(map.containsKey(x))
            {
                arr[0]=map.get(x);
                arr[1]=i;
                break;
            }
            map.put(nums[i],i);
        }
        return arr;
    }
}
```

-