Assignment 1 Answers - Arrays | DSA

Answer 1)

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

        int[] result=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){
                    result[0]=i;
                    result[1]=j;
                    break;
                }
            }
        }
        return result;
    }
}
```

Answer 2)

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

Answer 3)

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        int start=0;
        int end=nums.length-1;
        while(start<=end){
            int mid=(start+end)/2;
            if(nums[mid]==target){
                return mid;
            }else if(nums[mid]>target){
                end=mid-1;
            }else{
                start=mid+1;
            }
        }
        return start;
    }
}
```

Answer 4)

```
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;
    }
}
```

Answer 5)

```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        for(int i=0;i<n;i++){
            nums1[m+i]=nums2[i];
        }
        Arrays.sort(nums1);
    }
}
```

Answer 6)

```
class Solution {
    public boolean containsDuplicate(int[] nums) {
        HashMap<Integer,Integer> map = new HashMap<>();
        for (int i = 0; i < nums.length; i++) {
            if (map.containsKey(nums[i])) {
                return true;
            }
            map.put(nums[i],1);
        }
        return false;
    }
}
```

Answer 7)

```
class Solution {
    public void moveZeroes(int[] nums) {
        if(nums.length>1){
             int i=0;
             int j=1;
            while(i<nums.length && j<nums.length){
                if(nums[i]==0 && nums[j]!=0){
                    int temp=nums[i];
                    nums[i]=nums[j];
                    nums[j]=temp;
                    i++;
                    j++;
                }else if(nums[i]==0 && nums[j]==0){
                    j++;
                }else if(nums[i]!=0 && nums[j]!=0){
                    i++;
                    j++;
                }else if(nums[i]!=0 && nums[j]==0){
                    i++;
                    j++;
                }
            }
        }

    }
}
```

Answer 8)

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

        int[] res=new int[2];

        HashSet<Integer> set= new HashSet<>();
        int sum=0;
        for(int num: nums){
            if(set.contains(num)){
                res[0]=num;
            }else {
                set.add(num);
                 sum+=num;
            }

        }

        int n=nums.length;
        res[1]=(n*(n+1)/2)-(sum);
        return res;
    }
}
```