MILESTONE 2

1. https://leetcode.com/problems/search-in-rotated-sorted-array

```
class Solution {
    public int search(int[] nums, int target) {
        int l,r,m;
        l=0;
        r=nums.length-1;
        while(l<=r)
        {
            m=(l+r)/2;
            if(target==nums[m])
                return m;
            else if(nums[m]>=nums[l])
            {
                if(target<=nums[m] && target>=nums[l])
                    r=m-1;
                else
                    l=m+1;
            }
            else
            {
                if(target>=nums[m] && target<=nums[r])
                    l=m+1;
                else
                    r=m-1;
            }
        }
        return -1;
    }
}
```

2. https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array

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

3. https://leetcode.com/problems/find-the-duplicate-number

```
class Solution {
    public int findDuplicate(int[] nums) {
        //using the floyd cycle detection to find the duplicate number as there is only one
//repeated number and using constant extra space
        int slow=0,fast=0;
        do{
            slow=nums[slow];
            fast=nums[nums[fast]];
        }while(slow!=fast);

        slow=0;
        while(slow!=fast){
            slow=nums[slow];
            fast=nums[fast];
        }
        return fast;
    }
}
```

4. https://leetcode.com/problems/squares-of-a-sorted-array

```
class Solution {
    public int[] sortedSquares(int[] nums) {
        int ans[]=new int[nums.length];
        for(int i=0;i<nums.length;i++)
        {
            ans[i]=nums[i]*nums[i];
        }
        Arrays.sort(ans);
        return ans;
    }
}
```

5. https://leetcode.com/problems/maximum-ice-cream-bars

```
class Solution {
    public int maxIceCream(int[] costs, int coins) {
        Arrays.sort(costs);
        int sum=0,count=0,ans=0;
        for(int i=0;i<costs.length;i++)
        {
            sum=sum+costs[i];
            count++;
            if(sum==coins)
                return count;
            if(sum>coins)
                return (count-1);
        }
        return count;
    }
}
```

6. https://leetcode.com/problems/longest-substring-without-repeating-characters

```
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int i,j;//two pointers
        i=j=0;
        int max=0;
        HashSet<Character> hashset=new HashSet();
        while(j<s.length())
        {
            if(!hashset.contains(s.charAt(j)))
            {
                hashset.add(s.charAt(j));
                j++;
                max=Math.max(hashset.size(),max);
            }
            else
            {
                hashset.remove(s.charAt(i));
                i++;
            }
        }
        return max;
    }
}
```

7. https://leetcode.com/problems/longest-consecutive-sequence

```
class Solution {
    public int longestConsecutive(int[] nums) {
        Set<Integer> hashSet=new HashSet<Integer>();
        for(int i=0;i<nums.length;i++)
        {
            hashSet.add(nums[i]);
        }
        int c1=0;
        for(int i=0;i<nums.length;i++)
        {
            if(!hashSet.contains(nums[i]-1))
            {
                int temp=nums[i];
                int c2=1;
                while(hashSet.contains(temp+1))
                {
                    temp=temp+1;
                    c2++;
                }
                c1=Math.max(c1,c2);
            }
        }
        return c1;
    }
}
```

8. https://leetcode.com/problems/majority-element

```
class Solution {
    public int majorityElement(int[] nums) {
        int majority=nums[0];
        int c=1;
        for(int i=1;i<nums.length;i++)
        {
            if(nums[i]==majority)
                c++;
            else
                c--;
            if(c==0)
            {
                majority=nums[i];
                c=1;
            }
        }
        int count=0;
        for(int i=0;i<nums.length;i++)
        {
            if(nums[i]==majority)
                count++;
        }
        if(count>(nums.length/2))
            return majority;
        else
            return 0;
    }
}
```