1. Kth Smallest element

class Solution {  
    public static int kthSmallest(int[] arr, int k) {  
         int min=0;  
        while(k!=0){  
             min=Integer.MAX\_VALUE;  
            int j=0;  
            for(int i=0; i<arr.length; i++){  
                if(arr[i]<min){  
                    min=arr[i];  
                    j=i;  
                }  
            }  
            arr[j]=Integer.MAX\_VALUE;  
            k--;  
        }  
        return min;  
    }  
}

2)Minimize height 2:

Solution:

class Solution {  
    int getMinDiff(int[] arr, int k) {  
        int n =arr.length;  
        if( n == 1){  
            return 0;  
        }  
        Arrays.sort(arr);  
          
        int ans = arr[n-1] - arr[0];  
        int min = arr[0] + k;  
        int max = arr[n-1] - k;  
          
        for(int i=0; i<n-1; i++){  
            int min\_ele = Math.min(min, arr[i+1]-k);  
            int max\_ele = Math.max(max, arr[i]+k);  
                if(min\_ele < 0)  
                continue;  
            ans = Math.min(ans, max\_ele - min\_ele);  
        }  
        return ans;  
    }  
}

3)Parenthesis checker:

Solution:

static boolean isParenthesisBalanced(String s) {  
        Stack<Character> stk = new Stack<>();  
        for(char ch : s.toCharArray()) {  
              
            if(ch == '(') {  
                stk.push(')');  
            } else if(ch == '{') {  
                stk.push('}');  
            } else if(ch == '[') {  
                stk.push(']');  
            } else {  
                if(stk.isEmpty() || stk.pop() != ch) {  
                    return false;  
                }  
            }  
        }  
        return stk.isEmpty();  
    }

4)Equilibrium point:

Solution:

class Solution {  
    public static int equilibriumPoint(int arr[]) {  
        int rsum = 0;  
        for(int i:arr) rsum += i;  
        int lsum = 0;  
        for(int i=0;i<arr.length;i++){  
            if(lsum == (rsum - lsum-arr[i])) return i+1;  
            lsum += arr[i];  
        }  
        return -1;  
    }  
}

5)Binary search:

Solution:

class Solution {  
    public int binarysearch(int[] arr, int k) {  
       int left=0,right=arr.length-1,mid=0;  
       for(;left<=right;){  
           mid=(left+right)/2;  
           if(arr[mid]>k)  
               right=mid-1;  
             
            if(arr[mid]<k)  
               left=mid+1;  
             
           if(arr[mid]==k)  
           return mid;  
       }  
        return -1;  
        }  
    }

6) Next greatest element:

Solution:

class Solution {  
    public ArrayList<Integer> nextLargerElement(int[] arr) {

        int n=arr.length;  
        ArrayList<Integer>list=new ArrayList<>(n);  
        for(int i=0;i<n;i++){  
            list.add(-1);  
        }  
          
        for(int i=0;i<n;i++){  
            for(int j=i+1;j<n;j++){  
                if(arr[j]>arr[i]){  
                    list.set(i,arr[j]);  
                    break;  
                }  
            }  
        }  
        return list;  
    }  
}

7) Union of 2 array with duplicate element:

Solution:

class Solution {

    public static int findUnion(int a[], int b[]) {

        Set<Integer> set = new HashSet<Integer>();

        int max = Math.max(a.length,b.length);

        for(int i=0;i<max;i++){

            if(i<a.length)

               set.add(a[i]);

            if(i<b.length)

               set.add(b[i]);

        }

    return set.size();

}

}