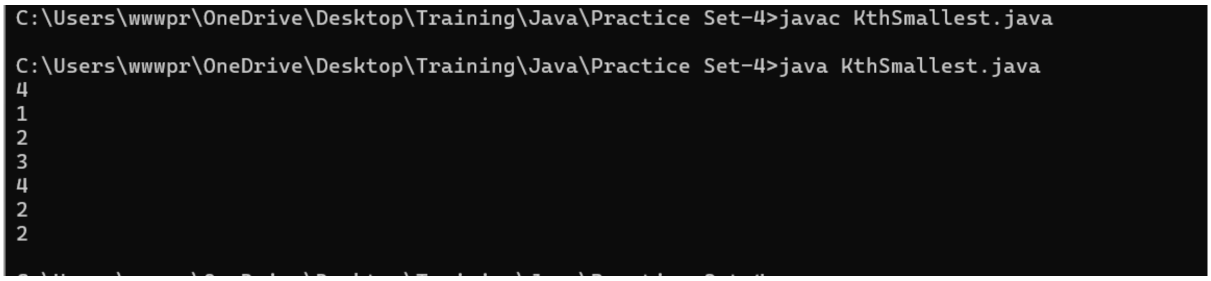
Name :Pradeep S
Dept   :CSBS

1)Kth Smallest

```
import java.util.*;

class KthSmallest{

    public static int Smallest(int[] arr, int k) {

        Arrays.sort(arr);

        return arr[k - 1];

    }

   public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        int k = sc.nextInt();

        int result = Smallest(arr, k);

        System.out.println(result);


    }

}
```



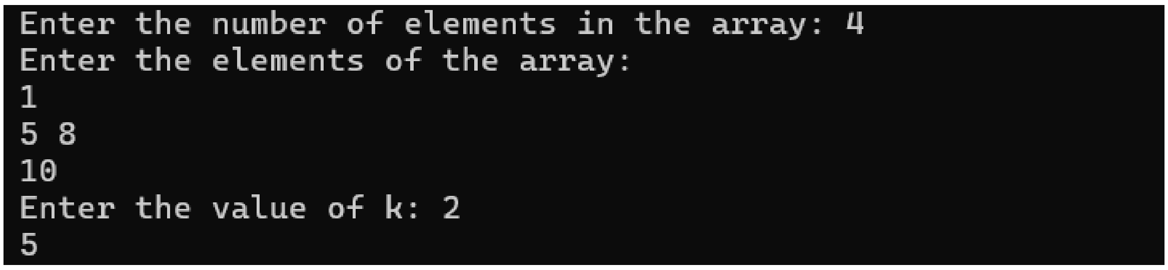
```
C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>javac KthSmallest.java

C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>java KthSmallest.java
4
1
2
3
4
2
2
```

Time Complexity : O(nlogn)

Space Complexity : O(n)

2)Minimum Height – II

```
import java.util.*;


class Solution {
    static int getMin(int[] arr, int k) {
        int n = arr.length;
        Arrays.sort(arr);


        int a = arr[0] + k;
        int b = arr[n - 1] - k;
        int ans = arr[n - 1] - arr[0];


        for (int i = 0; i < n - 1; i++) {
            int aa = Math.max(b, arr[i] + k);
            int bb = Math.min(a, arr[i + 1] - k);
            if (bb < 0) {
                continue;
            }
            ans = Math.min(ans, aa - bb);
        }

        return ans;
    }

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);

        System.out.print("Enter the number of elements in the array: ");
        int n = sc.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print("Enter the value of k: ");
        int k = sc.nextInt();
            System.out.println(getMin(arr,k));
    }
}
```



```
Enter the number of elements in the array: 4
Enter the elements of the array:
1
5 8
10
Enter the value of k: 2
5
```

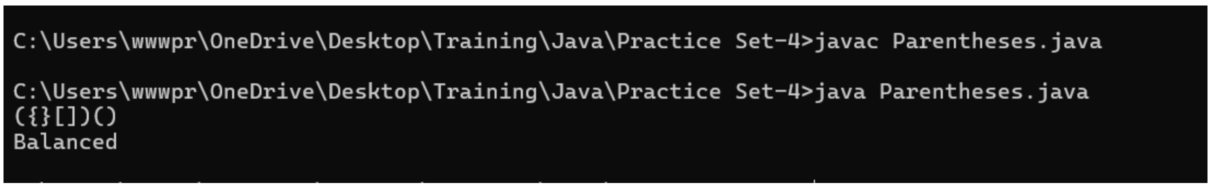
Time Complexity : O(n)

Space Complexity: O(n)

3)Valid Parentheses:

```
import java.util.*;

class Parentheses {
  static Boolean isBalanced(String s) {
    Stack<Character> a = new Stack<>();
    for (int i = 0; i < s.length(); i++) {
      char c = s.charAt(i);
      if (c == '(' || c == '{' || c == '[') {
        a.push(c);
      } else {
        if (a.isEmpty()) {
          return false;
        }
        char b = a.peek();
        if (c == ')' && b == '(' || c == '}' && b == '{' || c == ']' && b == '[') {
          a.pop();
        } else {
          return false;
        }
      }
    }
    return a.isEmpty();
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
```

```
        String s = sc.nextLine();

        boolean result = isBalanced(s);

        System.out.println(result ? "Balanced" : "Not Balanced");


    }

}
```



```
C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>javac Parentheses.java

C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>java Parentheses.java
({}[])()
Balanced
```

Time Complexity : O(n)

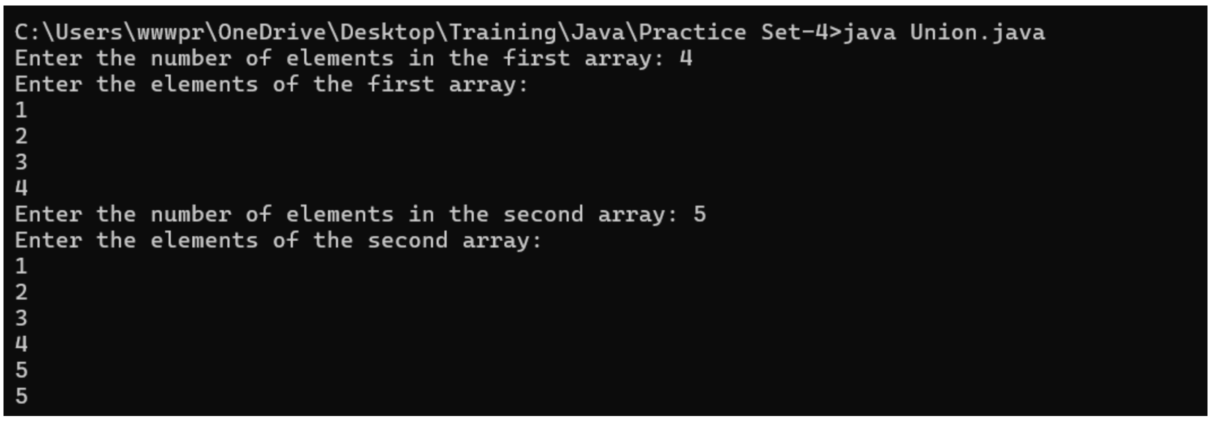
Space Complexity : O(n)

4)Union of two arrays without duplicates

```
import java.util.*;


class Union {

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

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

        int n = a.length;

        int m = b.length;

        int N = Math.max(n, m);


        for (int i = 0; i < N; i++) {

            if (i < n) {

                aa.add(a[i]);

            }

            if (i < m) {
```

```
        aa.add(b[i]);
      }
    }
    return aa.size();
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);


    System.out.print("Enter the number of elements in the first array: ");
    int n = sc.nextInt();
    int[] a = new int[n];
    System.out.println("Enter the elements of the first array: ");
    for (int i = 0; i < n; i++) {
      a[i] = sc.nextInt();
    }


    System.out.print("Enter the number of elements in the second array: ");
    int m = sc.nextInt();
    int[] b = new int[m];
    System.out.println("Enter the elements of the second array: ");
    for (int i = 0; i < m; i++) {
      b[i] = sc.nextInt();
    }
```

```
        int unionSize = findUnion(a, b);

        System.out.println(unionSize);




    }

}
```



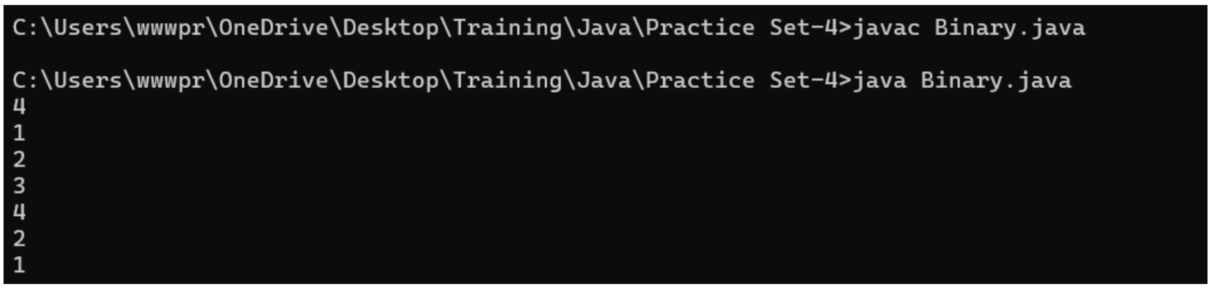
```
C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>java Union.java
Enter the number of elements in the first array: 4
Enter the elements of the first array:
1
2
3
4
Enter the number of elements in the second array: 5
Enter the elements of the second array:
1
2
3
4
5
5
```

Time Complexity : O(n)

Space Complexity : O(n)

5)Binary Search :

```
import java.util.*;


class Binary{

    public static int binarysearch(int[] arr, int k) {

        int l = 0;

        int r = arr.length - 1;

        while (l <= r) {

            int mid = (l + r) / 2;

            if (arr[mid] == k) return mid;

            else if (arr[mid] > k) r = mid - 1;

            else l = mid + 1;
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        int k = sc.nextInt();
        System.out.println(binarysearch(arr,k));

    }
}
```



```
C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>javac Binary.java

C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>java Binary.java
4
1
2
3
4
2
1
```

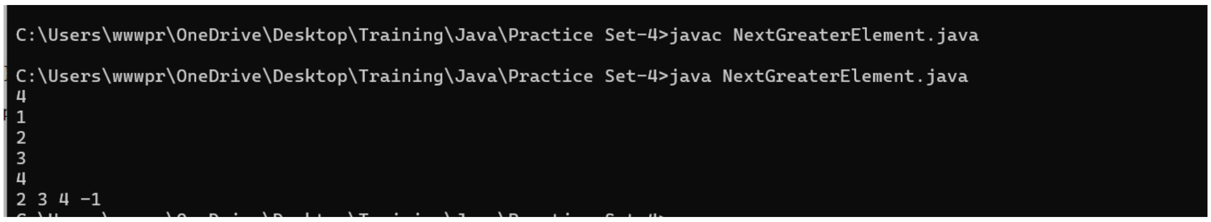
Time Complexity : O(logn)

Space Complexity : O(n)

6)Next Greater Element :

```
import java.util.*;
class NextGreaterElement {
    public static int[] nextGreaterElements(int[] arr) {
```

```
    int size = arr.length;
    int[] ans = new int[size];
    Stack<Integer> stack = new Stack<> ();

    for (int i = 2 * size - 1; i >= 0; i--) {
        while (!stack.isEmpty() && arr[i % size] >= stack.peek()) {
            stack.pop();
        }

        if (i < size) {
            if (!stack.isEmpty()) {
                ans[i] = stack.peek();
            } else {
                ans[i] = -1;
            }
        }

        stack.push(arr[i % size]);
    }

    return ans;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }



        int[] result = nextGreaterElements(arr);


        for (int value : result) {
            System.out.print(value + " ");
        }


    }
}
```



```
C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>javac NextGreaterElement.java

C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>java NextGreaterElement.java
4
1
2
3
4
2 3 4 -1
```

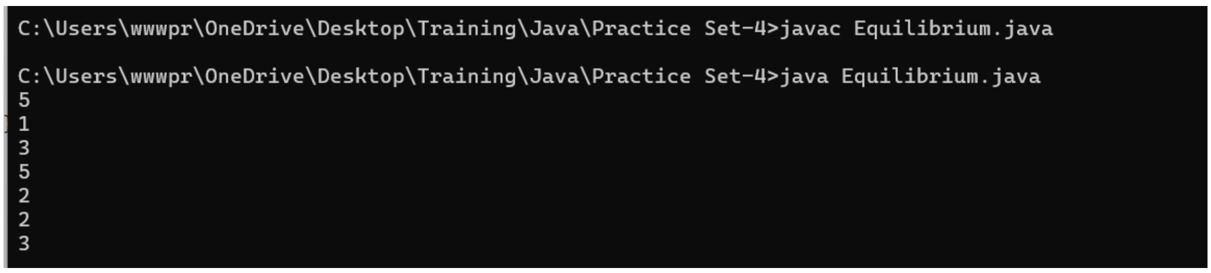
Time Complexity : O(n)

Space Complexity : O(n)

7)Equilbrium Points:

```
import java.util.*;


class Equilibrium {
    public static int equilibriumPoint(int arr[]) {
        int a = 0;
        for (int i : arr) a += i;
        int b = 0;

        for (int i = 0; i < arr.length; i++) {

            if (b == (a - b - arr[i])) return i + 1;

            b += arr[i];

        }

        return -1;

    }


    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        int result = equilibriumPoint(arr);

        System.out.println(result);


    }

}
```

Time Complexity : O(n)
Space Complexity : O(n)



```
C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>javac Equilibrium.java

C:\Users\wwwpr\OneDrive\Desktop\Training\Java\Practice Set-4>java Equilibrium.java
5
1
3
5
2
2
3
```