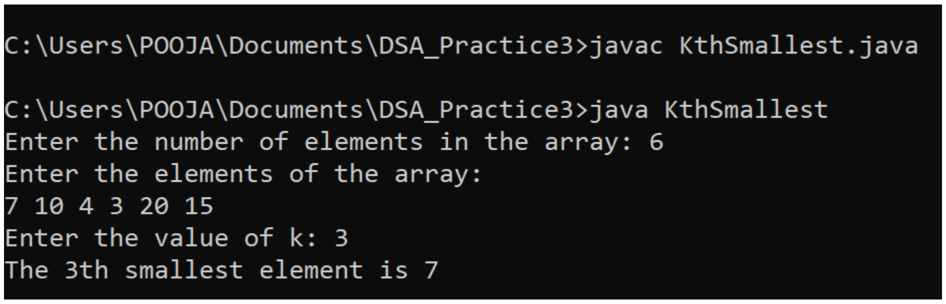
**12/11/2024 DSA PRACTICE 3**

**1.Kth Smallest Element**

```
import java.util.Arrays;

import java.util.Scanner;


public class KthSmallest{


    static int kthSmallest(int[] arr, int n, int k) {

        int max_element = arr[0];

        for (int i = 1; i < n; i++) {

            if (arr[i] > max_element) {

                max_element = arr[i];

            }

        }


        int[] freq = new int[max_element + 1];

        Arrays.fill(freq, 0);

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

            freq[arr[i]]++;

        }


        int count = 0;

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

            if (freq[i] != 0) {
                count += freq[i];
                if (count >= k) {
                    return i;
                }
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

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

        System.out.print("Enter the value of k: ");
        int k = scanner.nextInt();

        System.out.println("The " + k + "th smallest element is " + kthSmallest(arr, n, k));
        scanner.close();
    }
}
```

**Time Complexity** : O(n)

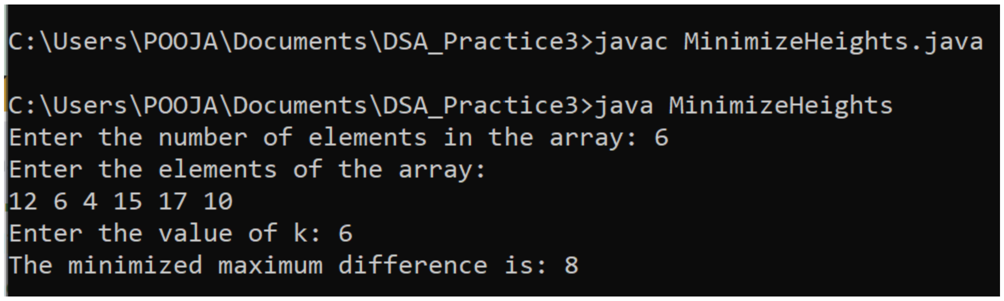


```
C:\Users\POOJA\Documents\DSA_Practice3>javac KthSmallest.java

C:\Users\POOJA\Documents\DSA_Practice3>java KthSmallest
Enter the number of elements in the array: 6
Enter the elements of the array:
7 10 4 3 20 15
Enter the value of k: 3
The 3th smallest element is 7
```

**2.Minimize The Heights**

```
import java.util.Arrays;

import java.util.Scanner;


class MinimizeHeights {


    static int getMinDiff(int[] arr, int k) {

        int n = arr.length;

        Arrays.sort(arr);

        int res = arr[n - 1] - arr[0];


        for (int i = 1; i < n; i++) {

            if (arr[i] - k < 0)

                continue;


            int minH = Math.min(arr[0] + k, arr[i] - k);

            int maxH = Math.max(arr[i - 1] + k, arr[n - 1] - k);

            res = Math.min(res, maxH - minH);
        }
        return res;
    }

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

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

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

        System.out.print("Enter the value of k: ");
        int k = scanner.nextInt();

        int ans = getMinDiff(arr, k);
        System.out.println("The minimized maximum difference is: " + ans);

        scanner.close();
    }
}
```

**Time Complexity** : O(nlogn)

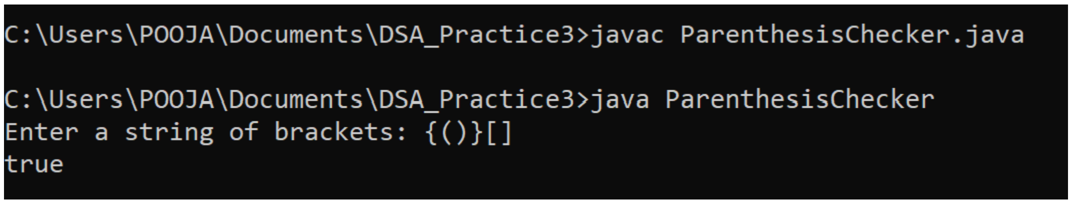


```
C:\Users\POOJA\Documents\DSA_Practice3>javac MinimizeHeights.java

C:\Users\POOJA\Documents\DSA_Practice3>java MinimizeHeights
Enter the number of elements in the array: 6
Enter the elements of the array:
12 6 4 15 17 10
Enter the value of k: 6
The minimized maximum difference is: 8
```

**3.Parenthesis Checker**

```
import java.util.Stack;

import java.util.Scanner;


public class ParenthesisChecker {

    public static boolean ispar(String s) {

        Stack<Character> stk = new Stack<>();

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

            if (s.charAt(i) == '(' || s.charAt(i) == '{' || s.charAt(i) == '[') {

                stk.push(s.charAt(i));

            } else {

                if (!stk.empty() &&

                    ((stk.peek() == '(' && s.charAt(i) == ')') ||

                     (stk.peek() == '{' && s.charAt(i) == '}') ||

                     (stk.peek() == '[' && s.charAt(i) == ']'))) {

                    stk.pop();

                } else {

                    return false;

                }

            }

        }
        return stk.empty();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string of brackets: ");
        String s = scanner.nextLine();

        if (ispar(s))
            System.out.println("true");
        else
            System.out.println("false");

        scanner.close();
    }
}
```

**Time Complexity :** O(n)

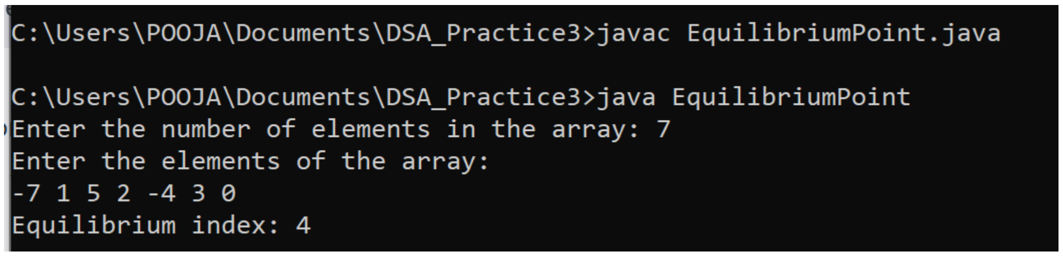


```
C:\Users\POOJA\Documents\DSA_Practice3>javac ParenthesisChecker.java

C:\Users\POOJA\Documents\DSA_Practice3>java ParenthesisChecker
Enter a string of brackets: {()}[]
true
```

**4.Equilibrium Point**

```
import java.util.Scanner;

public class EquilibriumPoint{

    public static int equilibriumPoint(long[] arr) {
        int n = arr.length;
        long leftsum, rightsum;

        for (int i = 0; i < n; ++i) {
            leftsum = 0;
            for (int j = 0; j < i; j++)
                leftsum += arr[j];

            rightsum = 0;
            for (int j = i + 1; j < n; j++)
                rightsum += arr[j];

            if (leftsum == rightsum)
                return i + 1;
        }
        return -1;
    }

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

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        long[] arr = new long[n];

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

        System.out.println("Equilibrium index: " + equilibriumPoint(arr));

        scanner.close();
    }
}
```

**Time Complexity** : O(n^2)



```
C:\Users\POOJA\Documents\DSA_Practice3>javac EquilibriumPoint.java

C:\Users\POOJA\Documents\DSA_Practice3>java EquilibriumPoint
Enter the number of elements in the array: 7
Enter the elements of the array:
-7 1 5 2 -4 3 0
Equilibrium index: 4
```

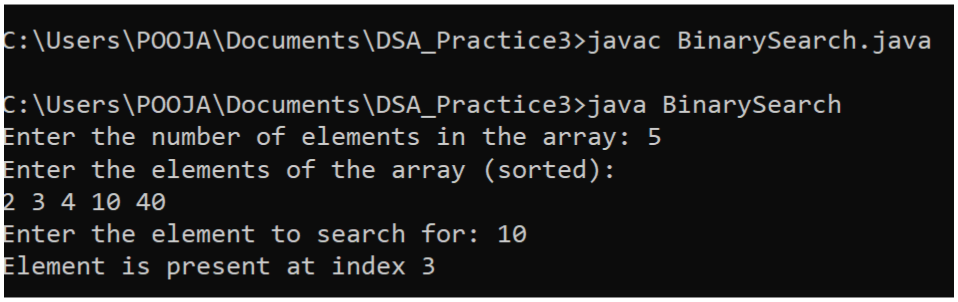
**5.Binary Search**

```
import java.util.Scanner;

class BinarySearch {

    int binarySearch(int arr[], int x) {
        int low = 0, high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;

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

            if (arr[mid] < x)
```

```
                low = mid + 1;
            else
                high = mid - 1;
        }

        return -1;
    }

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

        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

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

        System.out.print("Enter the element to search for: ");
        int x = scanner.nextInt();

        int result = ob.binarySearch(arr, x);
        if (result == -1)
            System.out.println("Element is not present in array");
        else
```

```
                System.out.println("Element is present at index " + result);


            scanner.close();

        }

}
```

**Time Complexity :** O(logn)

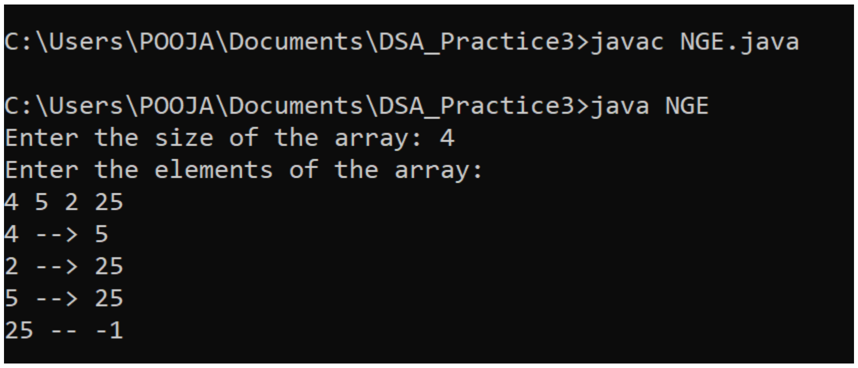


```
C:\Users\POOJA\Documents\DSA_Practice3>javac BinarySearch.java

C:\Users\POOJA\Documents\DSA_Practice3>java BinarySearch
Enter the number of elements in the array: 5
Enter the elements of the array (sorted):
2 3 4 10 40
Enter the element to search for: 10
Element is present at index 3
```

**6.Next Greater Element**

```
import java.util.Scanner;


public class NGE {

    static class stack {

        int top;

        int items[] = new int[100];


        void push(int x) {

            if (top == 99) {

                System.out.println("Stack full");

            } else {

                items[++top] = x;

            }

        }
        int pop() {
                if (top == -1) {
                        System.out.println("Underflow error");
                        return -1;
                } else {
                        int element = items[top];
                        top--;
                        return element;
                }
        }

        boolean isEmpty() {
                return (top == -1);
        }
}

static void printNGE(int arr[], int n) {
        int i = 0;
        stack s = new stack();
        s.top = -1;
        int element, next;

        s.push(arr[0]);

        for (i = 1; i < n; i++) {
                next = arr[i];
                if (!s.isEmpty()) {
                    element = s.pop();

                    while (element < next) {
                        System.out.println(element + " --> " + next);
                        if (s.isEmpty())
                            break;
                        element = s.pop();
                    }

                    if (element > next)
                        s.push(element);
                }

                s.push(next);
            }

            while (!s.isEmpty()) {
                element = s.pop();
                next = -1;
                System.out.println(element + " -- " + next);
            }
        }

        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter the size of the array: ");
            int n = scanner.nextInt();
            int[] arr = new int[n];

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

            printNGE(arr, n);
            scanner.close();
        }
    }
```

**Time Complexity :** O(n)

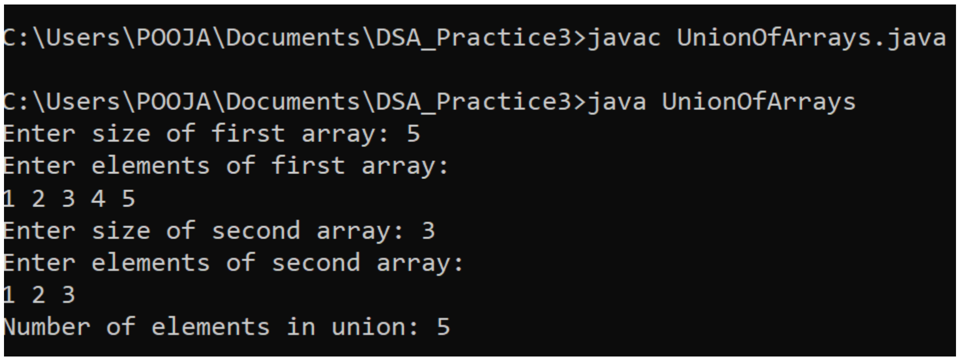


```
C:\Users\POOJA\Documents\DSA_Practice3>javac NGE.java

C:\Users\POOJA\Documents\DSA_Practice3>java NGE
Enter the size of the array: 4
Enter the elements of the array:
4 5 2 25
4 --> 5
2 --> 25
5 --> 25
25 -- -1
```

**7.Union of two arrays with duplicate elements**

```
import java.util.*;

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

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

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

        Set<Integer> unionSet = new HashSet<>();
        for (int value : a) {
            unionSet.add(value);
        }
        for (int value : b) {
            unionSet.add(value);
        }

        System.out.println("Number of elements in union: " + unionSet.size());
    }
}
```

**Time Complexity :** O(n+m)



```
C:\Users\POOJA\Documents\DSA_Practice3>javac UnionOfArrays.java

C:\Users\POOJA\Documents\DSA_Practice3>java UnionOfArrays
Enter size of first array: 5
Enter elements of first array:
1 2 3 4 5
Enter size of second array: 3
Enter elements of second array:
1 2 3
Number of elements in union: 5
```