Day-4 Practice SUM

Name: *Rishi Aathithya CN*

Reg no:*22IT086*

1.Kth Smallest element

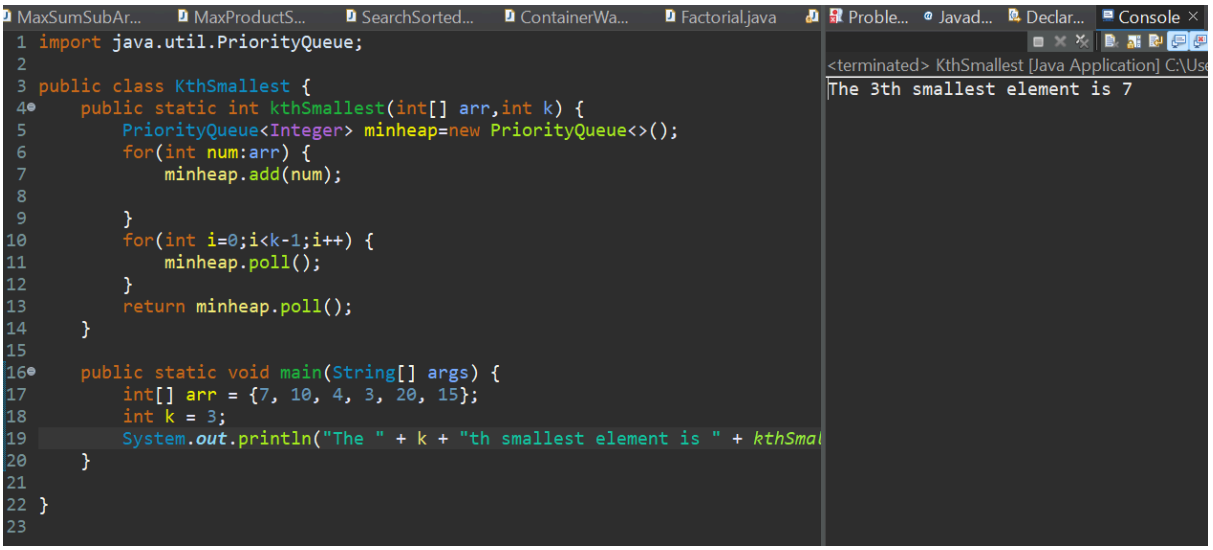
Solution:

```
import java.util.PriorityQueue;
public class KthSmallest {
        public static int kthSmallest(int[] arr,int k) {
                PriorityQueue<Integer> minheap=new PriorityQueue<>();
                for(int num:arr) {
                        minheap.add(num);

                }
                for(int i=0;i<k-1;i++) {
                        minheap.poll();
                }
                return minheap.poll();
        }

        public static void main(String[] args) {
    int[] arr = {7, 10, 4, 3, 20, 15};
    int k = 3;
    System.out.println("The " + k + "th smallest element is " + kthSmallest(arr, k));
  }
}
```

Output:




```
import java.util.PriorityQueue;

public class KthSmallest {
    public static int kthSmallest(int[] arr,int k) {
        PriorityQueue<Integer> minheap=new PriorityQueue<>();
        for(int num:arr) {
            minheap.add(num);

        }
        for(int i=0;i<k-1;i++) {
            minheap.poll();
        }
        return minheap.poll();
    }

    public static void main(String[] args) {
        int[] arr = {7, 10, 4, 3, 20, 15};
        int k = 3;
        System.out.println("The " + k + "th smallest element is " + kthSmal
    }

}
```

Time complexity: O(nlogn)

Space complexity: O(n)

2.Minimize the height II

Solution:

```
import java.util.Arrays;

public class MinHeightDifference {

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

    int n = arr.length;

    if (n == 1) {

      return 0;

    }

    Arrays.sort(arr);

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


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

      if (arr[i] - k < 0) continue;


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

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


      mindiff = Math.min(mindiff, max - min);
```

```
        }

        return mindiff;
    }

    public static void main(String[] args) {
        int[] arr = {1, 5, 8, 10};
        int k = 2;
        System.out.println("The minimum difference is: " + getMinDiff(arr, k));
    }
}
```

Output:

```
import java.util.Arrays;

public class MinHeightDifference {

    public static int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        if (n == 1) {
            return 0;
        }

        Arrays.sort(arr);

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

        for (int i = 1; i < n; i++) {
            if (arr[i] - k < 0) continue;

            int min = Math.min(arr[0] + k, arr[i] - k);
            int max = Math.max(arr[i - 1] + k, arr[n - 1] - k);

            mindiff = Math.min(mindiff, max - min);
        }

        return mindiff;
    }

    public static void main(String[] args) {
        int[] arr = {1, 5, 8, 10};
        int k = 2;
        System.out.println("The minimum difference is: " + getMinDiff(arr, k));
    }
}
```

```
<terminated> MinHeightDifference [Java Applic
The minimum difference is: 5
```

Time complexity: O(nlog n)

Space complexity: O(n)

3.Parenthesis Checker

Solution:

```
import java.util.Stack;
```

```
class ParanthesisCheck{

    static boolean isParenthesisBalanced(String s) {
        Stack<Character> st=new Stack<>();
        for(char it:s.toCharArray()){
            if(it=='(' || it=='[' || it=='{'){
                st.push(it);

            }
            else{
                if(st.isEmpty()){
                    return false;
                }
                char ch=st.pop();
                if(ch=='(' && it==')' ||
                ch=='[' && it==']' ||
                ch=='{' && it=='}'){
                    continue;
                }
                return false;
            }

        }
        return st.isEmpty();
    }
    public static void main (String args[]) {
            String s="(){}[]";
            System.out.println(isParenthesisBalanced(s));
    }
}
```

Output:

```java
import java.util.Stack;

class ParanthesisCheck{

    static boolean isParenthesisBalanced(String s) {
        Stack<Character> st=new Stack<>();
        for(char it:s.toCharArray()){
            if(it=='(' || it=='[' || it=='{'){
                st.push(it);

            }
            else{
                if(st.isEmpty()){
                    return false;
                }
                char ch=st.pop();
                if(ch=='(' && it==')' ||
                ch=='[' && it==']' ||
                ch=='{' && it=='}'){
                    continue;
                }
                return false;
            }

        }
        return st.isEmpty();
    }
    public static void main (String args[]) {
        String s="(){}[]";
        System.out.println(isParenthesisBalanced(s));
    }
}
```

```
<terminated> ParanthesisCheck [Java A
true
```

Time complexity: O(n)

Space complexity: O(n)

4.Equilibrium Point

Solution:

```
class EquillibriumPoint {


  public static int equilibriumPoint(int arr[]) {

    int n=arr.length;

    if(n==1){

      return 1;

    }

    int totalsum=0;

    for(int num:arr){

      totalsum+=num;
```

```
        }
        int left=0;
        for(int i=0;i<n;i++){
            int right=totalsum-left-arr[i];
            if(left==right){
                return i+1;
            }
            left+=arr[i];
        }
        return -1;

    }
    public static void main(String args[]) {
            int[] arr= {1,3,5,2,2};
            System.out.println("The equilibrium point is:"+equilibriumPoint(arr));
    }
}
```

Output:

```
class EquillibriumPoint {

    public static int equilibriumPoint(int arr[]) {
        int n=arr.length;
        if(n==1){
            return 1;
        }
        int totalsum=0;
        for(int num:arr){
            totalsum+=num;

        }
        int left=0;
        for(int i=0;i<n;i++){
            int right=totalsum-left-arr[i];
            if(left==right){
                return i+1;
            }
            left+=arr[i];
        }
        return -1;

    }
    public static void main(String args[]) {
        int[] arr= {1,3,5,2,2};
        System.out.println("The equilibrium point is:"+equilibriumPoint(arr));
    }
}
```

```
<terminated> EquillibriumPoint [Java Application]
The equilibrium point is:3
```

Time complexity: O(n)

Space complexity: O(1)

5.Binary Search

Solution:

```
public class BinarySearch {
    public static int binarySearch(int[] arr, int k) {
        int left = 0;
        int right = arr.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == k) {
                return mid;
            } else if (k < arr[mid]) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] arr = {2, 3, 5, 7, 8, 9};
        int k = 5;
        int result = binarySearch(arr, k);
        if (result != -1) {
            System.out.println("The target element is found at index: " + result);
        } else {
            System.out.println("The target element is not found in the array.");
        }
    }
}
```

Output:

```java
public class BinarySearch {
    public static int binarySearch(int[] arr, int k) {
        int left = 0;
        int right = arr.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == k) {
                return mid;
            } else if (k < arr[mid]) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] arr = {2, 3, 5, 7, 8, 9};
        int k = 5;
        int result = binarySearch(arr, k);
        if (result != -1) {
            System.out.println("The target element is found at inc
        } else {
            System.out.println("The target element is not found ir
        }
    }
}
```

```
<terminated> BinarySearch [Java Application] C:\Users\shan\.p
The target element is found at index: 2
```

Time complexity: O(log n)

Space complexity: O(1)

6.Next Greater Element

Solution:

```
import java.util.Scanner;

import java.util.Stack;


public class NextGreaterElement {

        public static void nextGreater(int[] arr) {

                Stack<Integer> stack=new Stack<>();

                int[] nge=new int[arr.length];


                for(int i=arr.length-1;i>=0;i--) {

                        while (!stack.isEmpty() && stack.peek() <= arr[i]) {

                                stack.pop();

                        }

                        nge[i] = stack.isEmpty() ? -1 : stack.peek();
```

```
                stack.push(arr[i]);
            }
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i] + " --> " + nge[i]);
        }
    }
```

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the elements of the array separated by spaces:");
    String input = scanner.nextLine();
    String[] elements = input.split(" ");
    int[] arr = new int[elements.length];
    for (int i = 0; i < elements.length; i++) {
        arr[i] = Integer.parseInt(elements[i]);
    }
    System.out.println("Next Greater Elements:");
   nextGreater(arr);

}
}
```

Output:

```java
import java.util.Scanner;
import java.util.Stack;

public class NextGreaterElement {
    public static void nextGreater(int[] arr) {
        Stack<Integer> stack=new Stack<>();
        int[] nge=new int[arr.length];

        for(int i=arr.length-1;i>=0;i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }
            nge[i] = stack.isEmpty() ? -1 : stack.peek();
            stack.push(arr[i]);
        }
         for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i] + " --> " + nge[i]);
        }
        }



public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the elements of the array separated b
    String input = scanner.nextLine();
    String[] elements = input.split(" ");
    int[] arr = new int[elements.length];
    for (int i = 0; i < elements.length; i++) {
        arr[i] = Integer.parseInt(elements[i]);
    }
    System.out.println("Next Greater Elements:");
   nextGreater(arr);
```

```
<terminated> NextGreaterElement [Java Application] C:\User
Enter the elements of the array separated b
4 5 2 25
Next Greater Elements:
4 --> 5
5 --> 25
2 --> 25
25 --> -1
```

Time complexity: O(n)

Space complexity: O(n)

7.Union of Two Array with Duplicate

Solution:

```java
import java.util.HashSet;


public class UnionTwoArray {

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

                HashSet<Integer> hash=new HashSet<>();

                for(int num:a) {

                        hash.add(num);

                }

                for(int num:b) {

                        hash.add(num);

                }

                return hash.size();

                }
```

```
        public static void main(String[] args) {
                int[] a= {1,2,3,4,5};
                int[] b= {1,2,3,6};
                System.out.println("The Union of two Array with duplicate is:"+unionTwoArray(a,b));



        }


}
```

Output:

```
import java.util.HashSet;

public class UnionTwoArray {
    public static int unionTwoArray(int[] a,int[] b) {
        HashSet<Integer> hash=new HashSet<>();
        for(int num:a) {
            hash.add(num);
        }
        for(int num:b) {
            hash.add(num);
        }
        return hash.size();
        }

    public static void main(String[] args) {
        int[] a= {1,2,3,4,5};
        int[] b= {1,2,3,6};
        System.out.println("The Union of two Array with duplicate


    }

}
```

```
<terminated> UnionTwoArray [Java Application] C:\Users\shan\.p
The Union of two Array with duplicate is:6
```

Time complexity: O(n+m)

Space complexity: O(n+m)