**Sanjay S,22IT093  
DSA PRACTICE SET 4**

**1.Kth Smallest**

import java.util.\*;

class Solution {

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

PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> a - b);

for (int num : arr) {

pq.add(num);

}

if (pq.size() < k) return -1;

int crr = 1;

while (crr++ < k) {

pq.remove();

}

return pq.remove();

}

public static void main(String[] args) {

Solution obj = new Solution();

int[] arr = {12, 3, 5, 7, 19};

int k = 4;

System.out.println("The " + k + "th smallest element is " + obj.kthSmallest(arr, k));

}

}



**2. Minimize the Heights II**

import java.util.Arrays;

class Solution {

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

Arrays.sort(arr);

int n = arr.length;

int res = 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);

res = Math.min(res, max - min);

}

return res;

}

public static void main(String[] args) {

Solution obj = new Solution();

int[] arr = {1, 5, 8, 10};

int k = 2;

System.out.println("The minimum difference is: " + obj.getMinDiff(arr, k));

}

}



**3.Parenthesis Checker :**

import java.util.\*;

public class BalancedParenthesis {

public static void main(String[] args) {

String str="())((())";

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

for (char c : str.toCharArray()) {

if (!st.isEmpty() && c == ')' && st.peek() == '(') {

st.pop();

} else {

st.push(c);

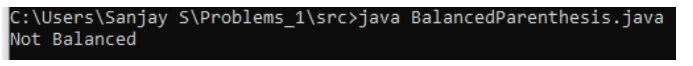
}

}

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

}

}



**4.Equilibrium Point**

class Solution {

public static int equilibriumPoint(int arr[]) {

// code here

int n=arr.length;

int[] prefixSum=new int[n+1];

prefixSum[0]=0;

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

prefixSum[i+1]=prefixSum[i]+arr[i];

}

int totSum=prefixSum[n];

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

if(prefixSum[i]==totSum-prefixSum[i+1]){

return i+1;

}

}

return -1;

}

public static void main(String[] args) {

Solution obj = new Solution();

int arr[] = {1, 3, 5, 2, 2};

System.out.println("Equilibrium point is: " + obj.equilibriumPoint(arr));

}

}

**5.Next Greater Element :**

import java.util.ArrayList;

import java.util.Stack;

class NextGreaterElement {

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

int n = arr.length;

ArrayList<Integer> ans = new ArrayList<>();

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

ans.add(-1);

}

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

for (int i = n - 1; i >= 0; i--) {

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

st.pop();

}

if (!st.isEmpty()) {

ans.set(i, st.peek());

}

st.push(arr[i]);

}

return ans;

}

public static void main(String[] args) {

int[] arr = {4 , 5 , 2 , 25};

ArrayList<Integer> result =nextLargerElement(arr);

System.out.println(result);

}

**6. Union of Arrays with Duplicates**

class Solution {

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

// code here

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

for(int num:a){

set.add(num);

}

for(int num:b){

set.add(num);

}

return set.size();

}

public static void main(String[] args) {

Solution obj = new Solution();

int a[] = {1, 2, 3, 4};

int b[] = {3, 4, 5, 6};

System.out.println("Union size: " + obj.findUnion(a, b));

}

}

