**Name: Abishek S H Date: 13/11/2024**

**Dept: CSE**

**1.Kth smallest element**

**Code:**

package sample4;

import java.util.\*;

public class kSmallest {

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

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

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

pq.offer(arr[i]);

if(pq.size()>k) {

pq.poll();

}

}

return pq.peek();

}

public static void main(String[] args) {

int[] ar= {1,28,5,79,23,12};

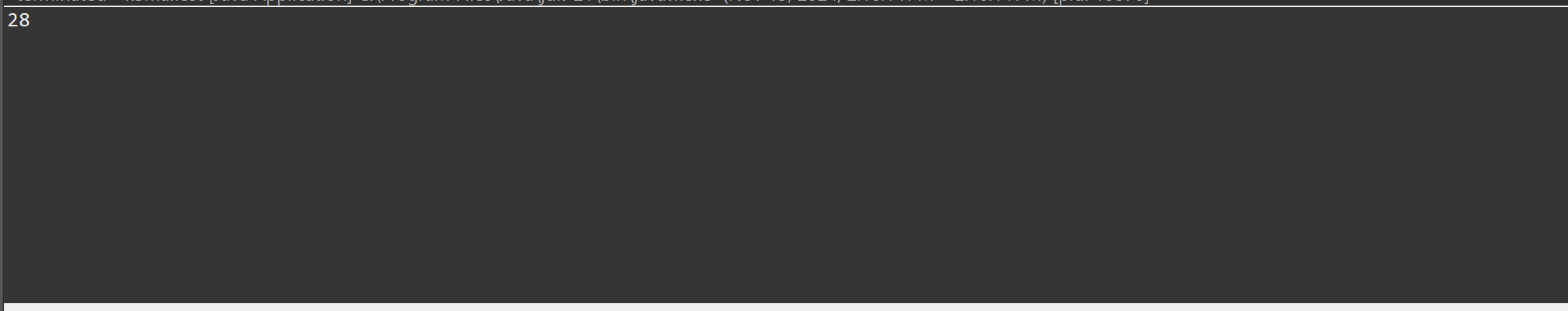
int res=*small*(ar,2);

System.***out***.println(res);

}

}

**Output:**



**Time Complexity:** O(n log k)

**2.Minimize the height**

**Code:**

package sample4;

import java.util.\*;

public class MinimizeHeight {

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

int n = arr.length;

if (n == 1) {

return 0;

}

Arrays.*sort*(arr);

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

int minHeight, maxHeight;

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

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

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

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

min = Math.*min*(min, maxHeight - minHeight);

}

return min;

}

public static void main(String[] args) {

int[] arr1 = {1, 5, 8, 10};

int k1 = 2;

System.***out***.println("Minimum difference for arr1: " + *diff*(arr1, k1)); // Output: 5

int[] arr2 = {3, 9, 12, 16, 20};

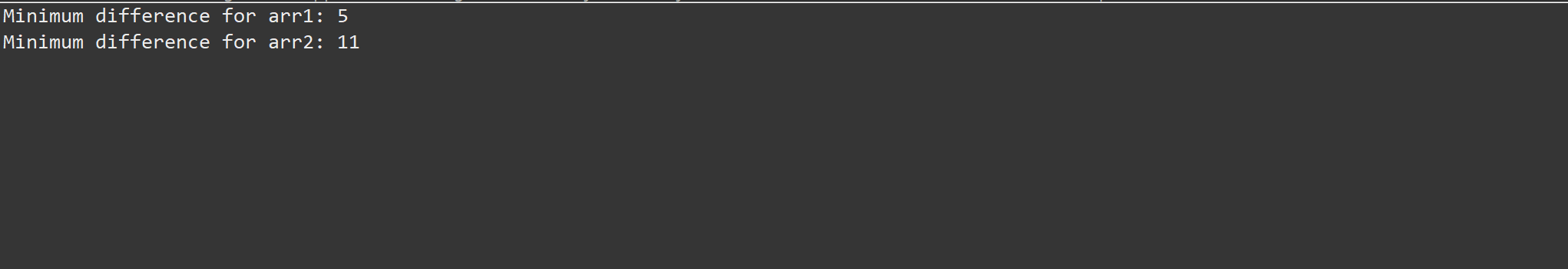
int k2 = 3;

System.***out***.println("Minimum difference for arr2: " + *diff*(arr2, k2)); // Output: 11

}

}

**Output:**



**Time Complexity:** O(n log n)

**3.Parenthesis checker**

**Code:**

package sample4;

import java.util.\*;

public class ValidParenthesis {

static boolean isValid(String s) {

boolean flag=true;

if (s.length() == 0) {

flag=false;

}

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

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

char c = s.charAt(i);

if (c == '(' || c == '{' || c == '[') {

st.push(c);

} else {

if (st.isEmpty()) {

flag=false;

break;

}

char ch = st.pop();

if ((c == ')' && ch != '(') || (c == '}' && ch != '{') || (c == ']' && ch != '[')) {

flag=false;

break;

}

}

}

return flag;

}

public static void main(String[] args) {

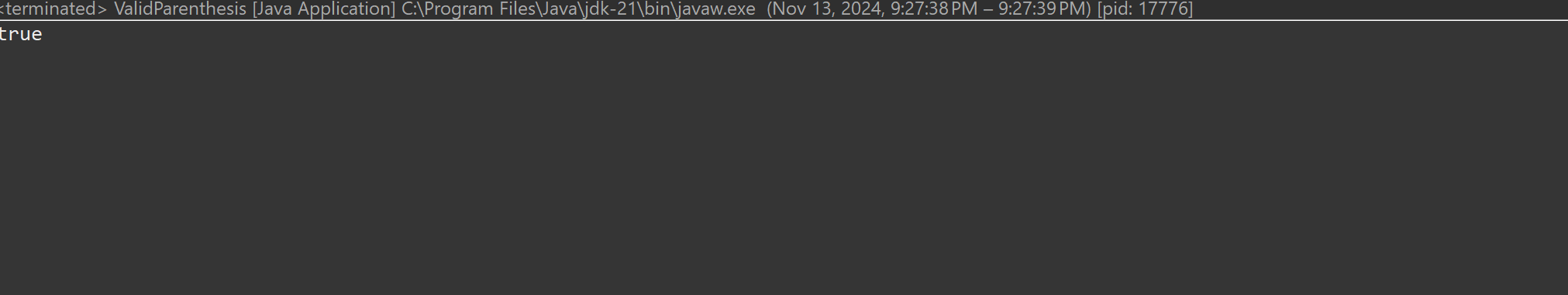
boolean f=*isValid*("{[()]}");

System.***out***.println(f);

}

}

**Output:**



**Time Complexity:** O(n)

**4. Equilibrium point**

**Code:**

package sample4;

import java.util.\*;

public class equil {

public static int findEquilibriumPoint(int[] arr) {

int n = arr.length;

if (n == 1) {

return 1;

}

int total = 0;

for (int num : arr) {

total += num;

}

int lSum = 0;

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

int rSum = total - arr[i] - lSum;

if (lSum == rSum) {

return i + 1;

}

lSum += arr[i];

}

return -1;

}

public static void main(String[] args) {

int[] arr1 = {1, 3, 5, 2, 2};

System.***out***.println("Equilibrium point in arr1: " + *findEquilibriumPoint*(arr1));

int[] arr2 = {1};

System.***out***.println("Equilibrium point in arr2: " + *findEquilibriumPoint*(arr2));

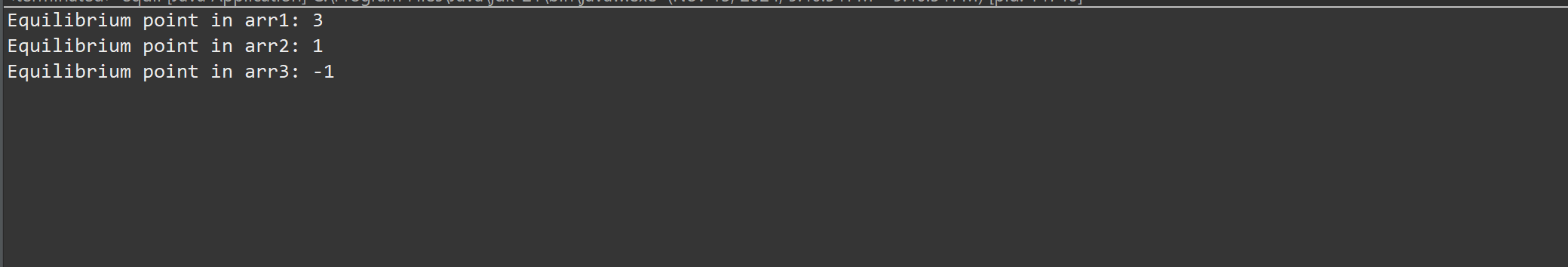
int[] arr3 = {1, 2, 3};

System.***out***.println("Equilibrium point in arr3: " + *findEquilibriumPoint*(arr3));

}

}

**Output:**

****

**Result:** O(n)

**5.Binary search**

**Code:**

package sample4;

import java.util.\*;

public class BinarySearch {

static int search(int[] arr,int target) {

Arrays.*sort*(arr);

int low=0;

int high=arr.length-1;

while(low<=high) {

int mid=low+(high-low)/2;

if(arr[mid]==target) {

return mid;

}

else if(arr[mid]<target) {

low=mid+1;

}

else {

high=mid-1;

}

}

return -1;

}

public static void main(String[] args) {

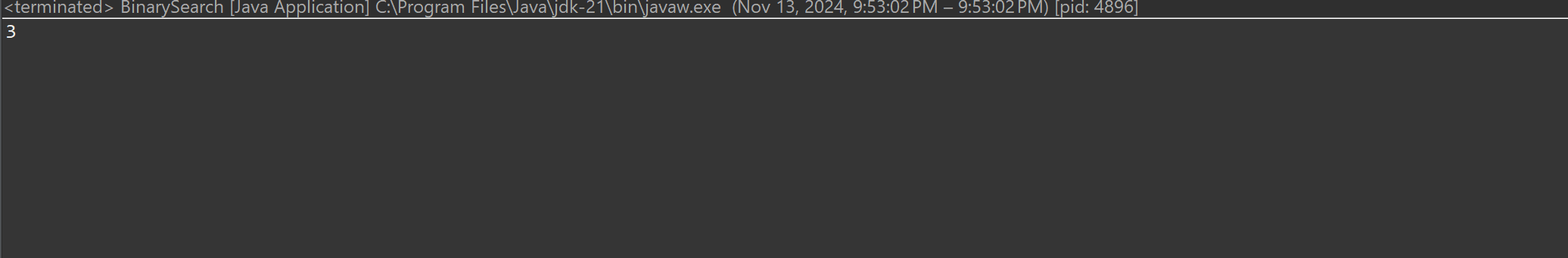
int[] ar= {8,9,10,12,25,36};

int res=*search*(ar,12);

System.***out***.println(res); }

}

**Output:**

****

**Time Complexity:** O(log n)

**6.Next Greater Element**

**Code:**

package sample4;

public class nextGreat {

static void solve(int arr[], int n)

{

int next, i, j;

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

next = -1;

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

if (arr[i] < arr[j]) {

next = arr[j];

break;

}

}

System.***out***.println(arr[i] + "->" + next);

}

}

public static void main(String[] args) {

int arr[] = { 11, 13, 21, 3 };

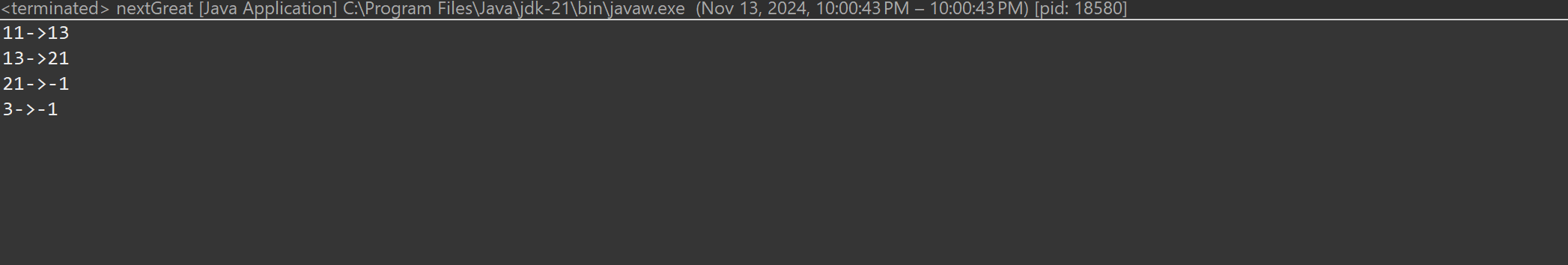
int n = arr.length;

*solve*(arr, n);

}

}

**Output:**

****

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

**7. union of two array with duplicate elements**

**Code:**

package sample4;

import java.util.\*;

public class UnionArray {

static int check(int[] arr1,int[] arr2) {

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

for (int num : arr1) {

hm.add(num);

} for (int num : arr2) {

hm.add(num);

}

return hm.size();

}

public static void main(String[] args) {

int[] a= {1,2,3,4,5};

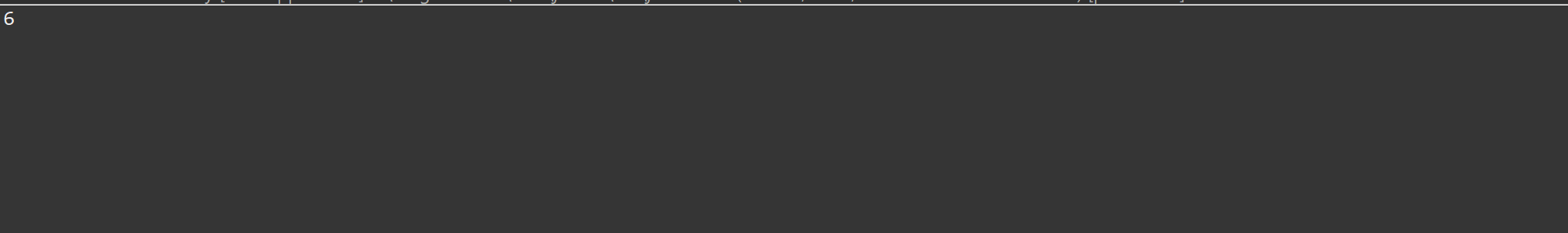
int[] b= {1,2,7,3};

System.***out***.println(*check*(a,b));

}

}

**Output:**



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