Name: Abishek S H Date:11/11/24

Dept: CSE

* + 1. **Knapsack Problem**

**Code:**

package sample2;

public class knapsock {

public static int knap(int W, int[] weights, int[] values, int n) {

int[] dp = new int[W + 1];

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

for (int w = W; w >= weights[i]; w--) {

dp[w] = Math.*max*(dp[w], values[i] + dp[w - weights[i]]);

}

}

return dp[W];

}

public static void main(String[] args) {

int[] values = {60, 100, 120};

int[] weights = {10, 20, 30};

int W = 50;

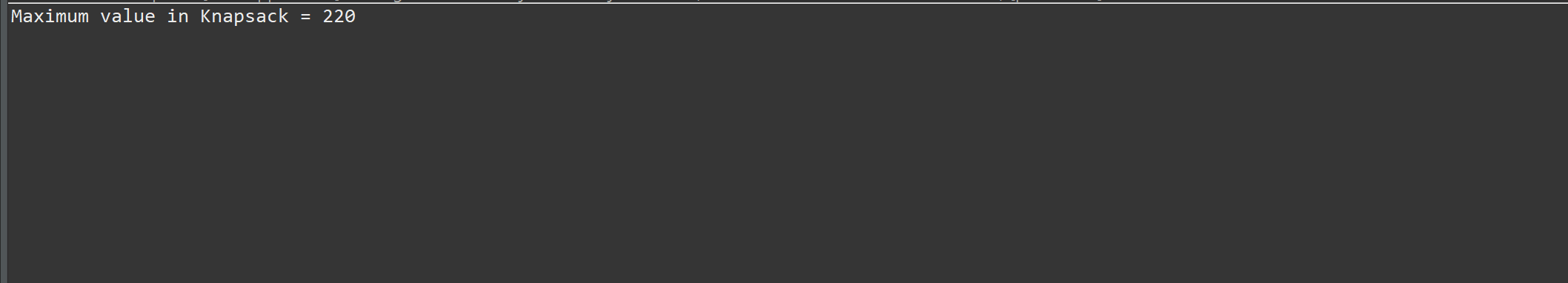
int n = values.length;

System.***out***.println("Maximum value in Knapsack = " + *knap*(W, weights, values, n));

}

}

**Output:**



**Time Complexity:** O(n\*W)

**2.** **Floor in sorted array**

**Code:**

package sample2;

public class floorIn{

public static int findFloor(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;

}

if (arr[mid] < k) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

public static void main(String[] args) {

int[] arr = {1, 3, 5, 7, 9, 11, 15};

int k = 7;

int result = *findFloor*(arr, k);

if (result != -1) {

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

} else {

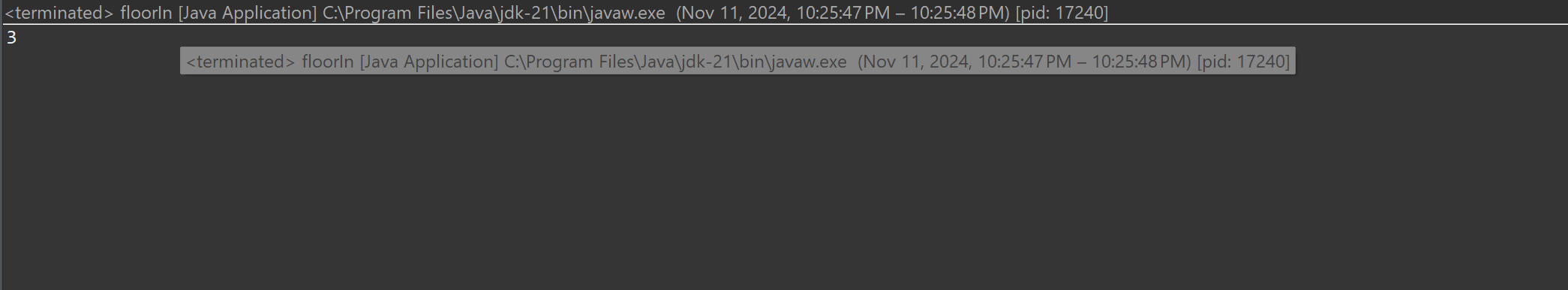
System.***out***.println(-1);

}

}

}

**Output:**



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

**3. Check equal arrays**

**Code:**

package sample2;

import java.util.\*;

public class checkArr {

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

if (arr1.length != arr2.length) {

return "Not Equal";

}

HashMap<Integer, Integer> map1 = new HashMap<>();

HashMap<Integer, Integer> map2 = new HashMap<>();

for (int num : arr1) {

map1.put(num, map1.getOrDefault(num, 0) + 1);

}

for (int num : arr2) {

map2.put(num, map2.getOrDefault(num, 0) + 1);

}

if (map1.equals(map2)) {

return "Equal";

}

return "Not Equal";

}

public static void main(String[] args) {

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

int[] arr2 = {1, 2, 2, 1};

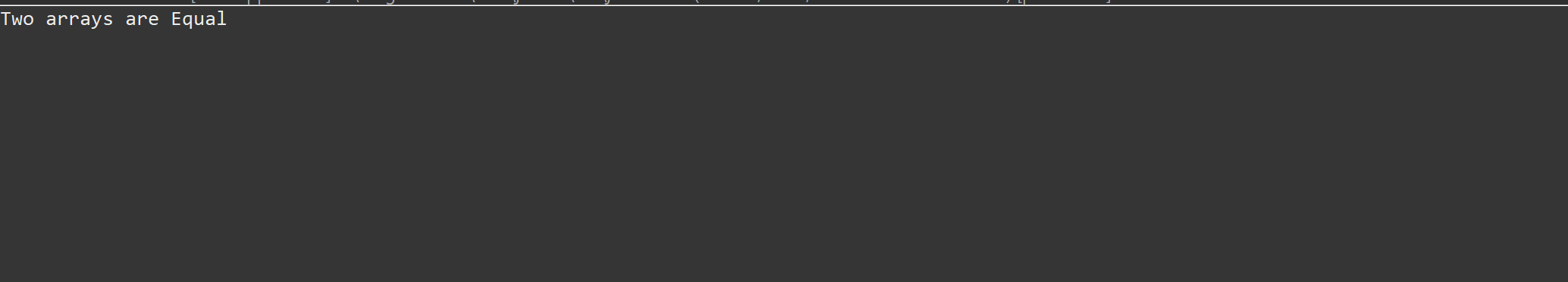
String res = *check*(arr1, arr2);

System.***out***.println("Two arrays are " + res);

}

}

**Output:**



**Time Complexity:** O(n)

**4.Linked List is palindrome or not**

**Code:**

package sample2;

import java.util.\*;

public class checkPal {

public static class ListNode {

int val;

ListNode next;

ListNode() {}

ListNode(int val) { this.val = val; }

ListNode(int val, ListNode next) { this.val = val; this.next = next; }

}

public ListNode findMid(ListNode head) {

ListNode slow = head;

ListNode fast = head;

while(fast !=null && fast.next != null ){

slow = slow.next;

fast = fast.next.next;

}

return slow;

}

public boolean isPalindrome(ListNode head) {

if(head == null && head.next != null) {

return true;

}

ListNode mid = findMid(head);

ListNode prev = null;

ListNode curr = mid;

ListNode next;

while(curr != null) {

next = curr.next;

curr.next = prev;

prev = curr;

curr = next;

}

ListNode right = prev;

ListNode left = head;

while(right != null) {

if(left.val != right.val) {

return false;

}

left = left.next;

right = right.next;

}

return true;

}

public static void main(String[] args) {

ListNode head = new ListNode(1);

head.next = new ListNode(2);

head.next.next = new ListNode(2);

head.next.next.next = new ListNode(1);

checkPal instance = new checkPal();

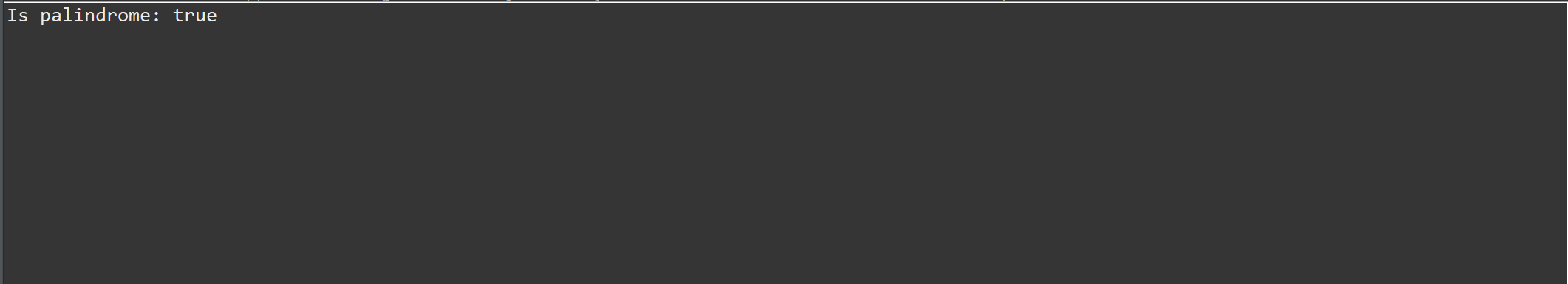
boolean result = instance.isPalindrome(head);

System.***out***.println("Is palindrome: " + result);

}

}

**Output:**



**Time Complexity:** O(n)

**5.Balanced Tree check**

**Code:**

package sample2;

import java.util.\*;

public class biBal {

public static class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode(int val) {

this.val = val;

left = right = null;

}

}

public static boolean isBalanced(TreeNode root) {

return *checkHeight*(root) != -1;

}

private static int checkHeight(TreeNode node) {

if (node == null) {

return 0;

}

int leftHeight = *checkHeight*(node.left);

if (leftHeight == -1) return -1;

int rightHeight = *checkHeight*(node.right);

if (rightHeight == -1) return -1;

if (Math.*abs*(leftHeight - rightHeight) > 1) {

return -1;

}

return Math.*max*(leftHeight, rightHeight) + 1;

}

public static void main(String[] args) {

TreeNode root = new TreeNode(10);

root.left = new TreeNode(20);

root.right = new TreeNode(30);

root.left.left = new TreeNode(40);

root.left.right = new TreeNode(60);

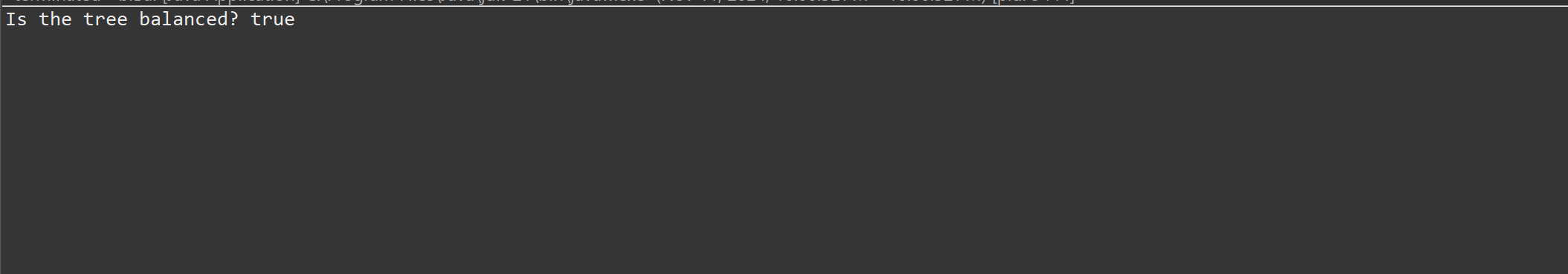
boolean result = *isBalanced*(root);

System.***out***.println("Is the tree balanced? " + result);

}

}

**Output:**

****

**Time Complexity:** O(n)

**6. Triplet Sum in array**

**Code:**

package sample2;

import java.util.Arrays;

public class tripletSum {

public static int[] findTriplet(int[] arr, int sum) {

Arrays.*sort*(arr);

for (int i = 0; i < arr.length - 2; i++) {

int left = i + 1;

int right = arr.length - 1;

while (left < right) {

int currentSum = arr[i] + arr[left] + arr[right];

if (currentSum == sum) {

return new int[] {arr[i], arr[left], arr[right]};

}

if (currentSum < sum) {

left++;

}

else {

right--;

}

}

}

return null;

}

public static void main(String[] args) {

int[] arr = {12, 3, 4, 1, 6, 9};

int sum = 24;

int[] triplet = *findTriplet*(arr, sum);

if (triplet != null) {

System.***out***.println(Arrays.*toString*(triplet));

} else {

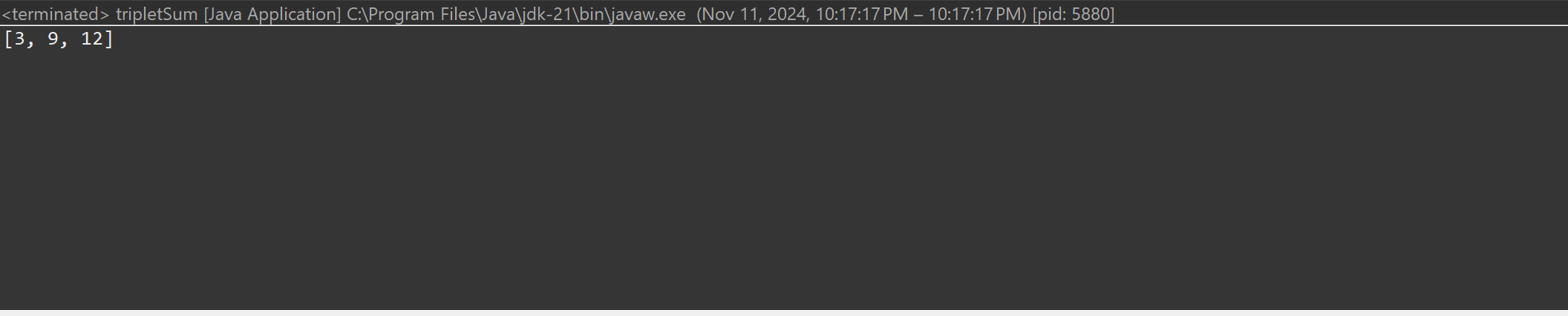
System.***out***.println(-1);

}

}

}

**Output:**



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