**Date: 11.11.2024**

**Sri Vaigunth M-22AD134**

**DSA Coding Problems**

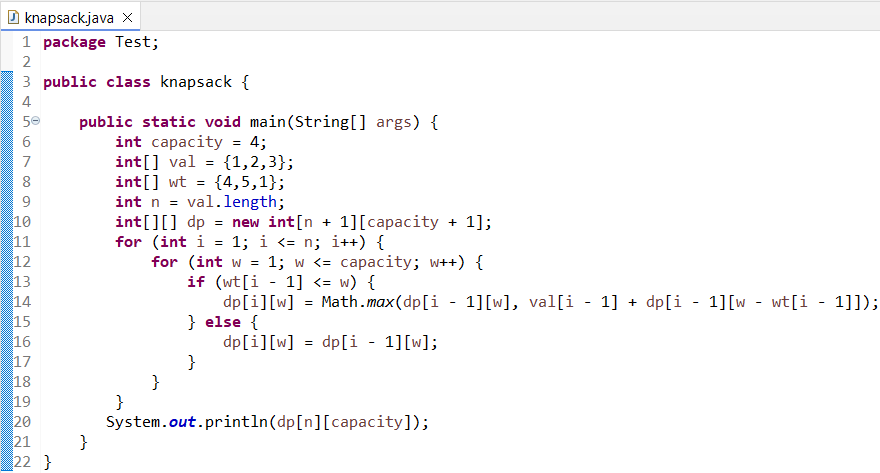
Set – 2

1. **0/1 Knapsack Problem:**

You are given the weights and values of items, and you need to put these items in a knapsack of capacity capacity to achieve the maximum total value in the knapsack. Each item is available in only one quantity. In other words, you are given two integer arrays val[] and wt[], which represent the values and weights associated with items, respectively. You are also given an integer capacity, which represents the knapsack capacity. Your task is to find the maximum sum of values of a subset of val[] such that the sum of the weights of the corresponding subset is less than or equal to capacity. You cannot break an item; you must either pick the entire item or leave it (0-1 property).

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

**Code:**

****

**Input:** capacity = 4, val[] = [1, 2, 3], wt[] = [4, 5, 1]

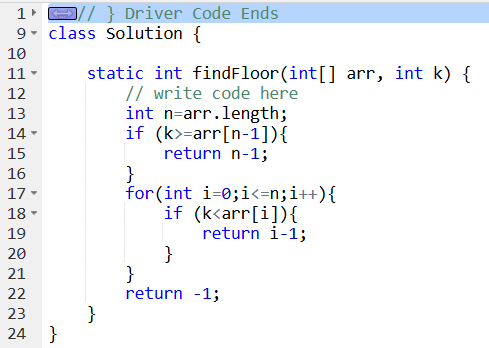
**Output:** 3

1. **Floor in sorted array:**

Given a sorted array and a value x, the floor of x is the largest element in the array smaller than or equal to x. Write efficient functions to find the floor of x .

**Time Complexity: O(N)**

**Code:**

****

**Input:** arr[] = [1, 2, 8, 10, 11, 12, 19], k = 5

**Output:** 1

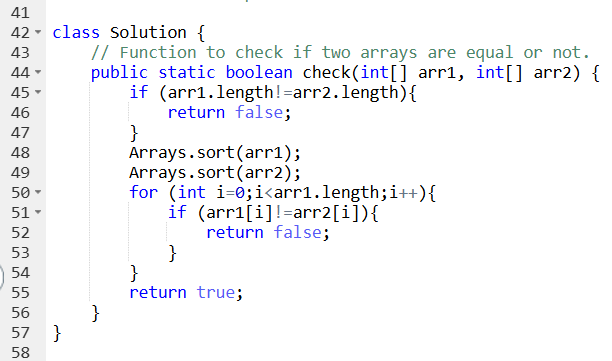
1. **Check equal arrays:**

Given two arrays arr1 and arr2 of equal size, the task is to find whether the given arrays are equal. Two arrays are said to be equal if both contain the same set of elements, arrangements (or permutations) of elements may be different though.

Note: If there are repetitions, then counts of repeated elements must also be the same for two arrays to be equal.

**Time Complexity: O(n)**

**Code:**

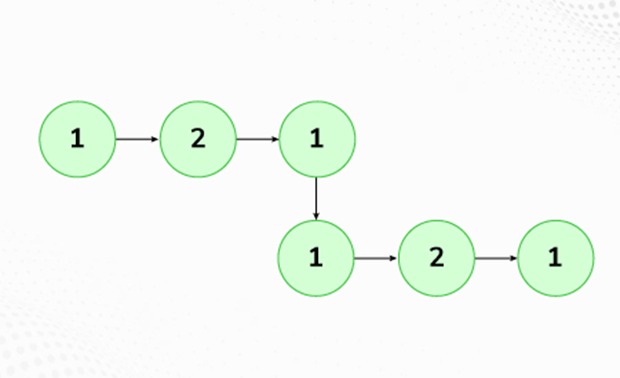
****

**Input:** arr1[] = [1, 2, 5, 4, 0], arr2[] = [2, 4, 5, 0, 1]

**Output:** true

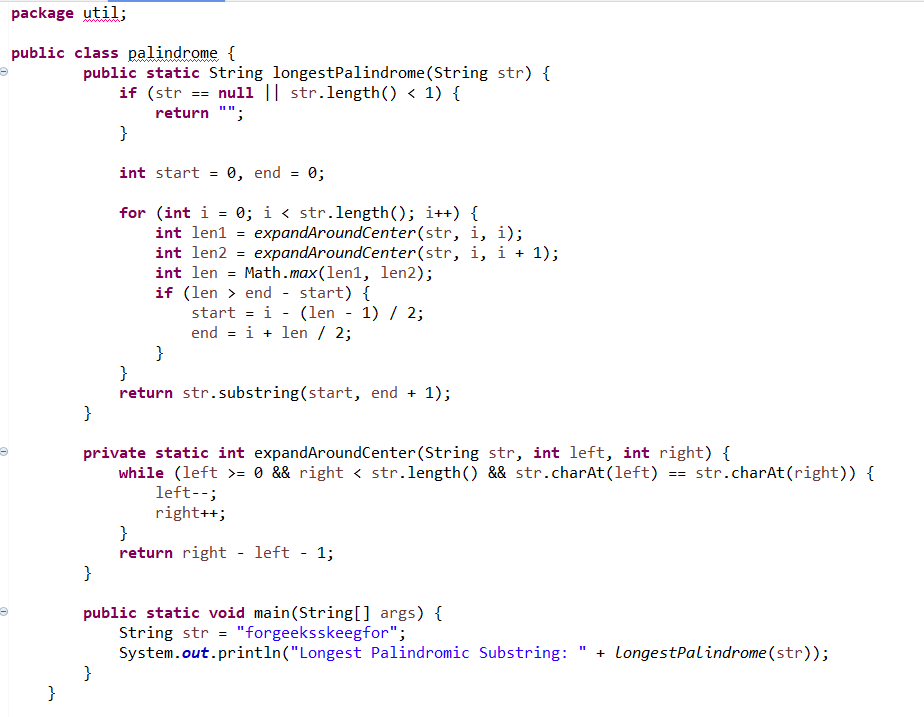
1. **Palindrome Linked List:**

Given a singly linked list of integers. The task is to check if the given linked list is palindrome or not.



**Time Complexity: O(n)**

**Code:**

****

**Input:** LinkedList: 1->2->1->1->2->1

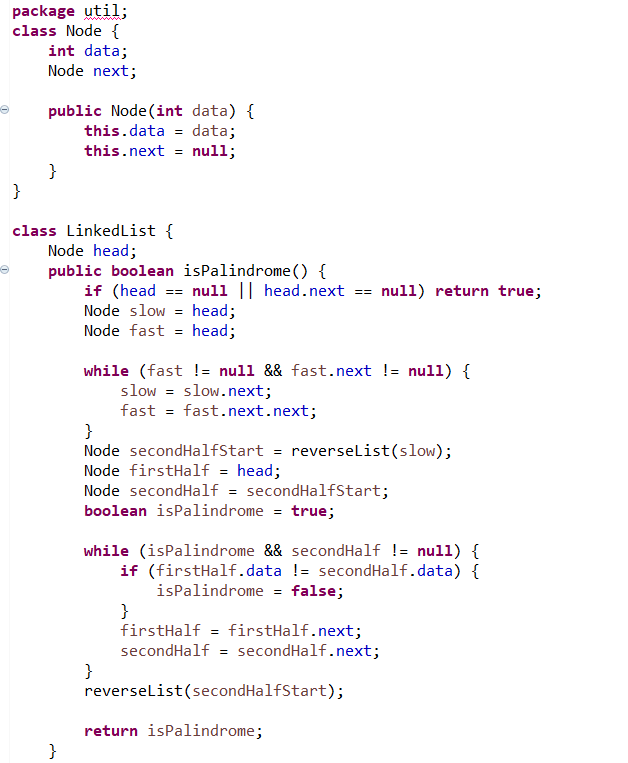
**Output:** true

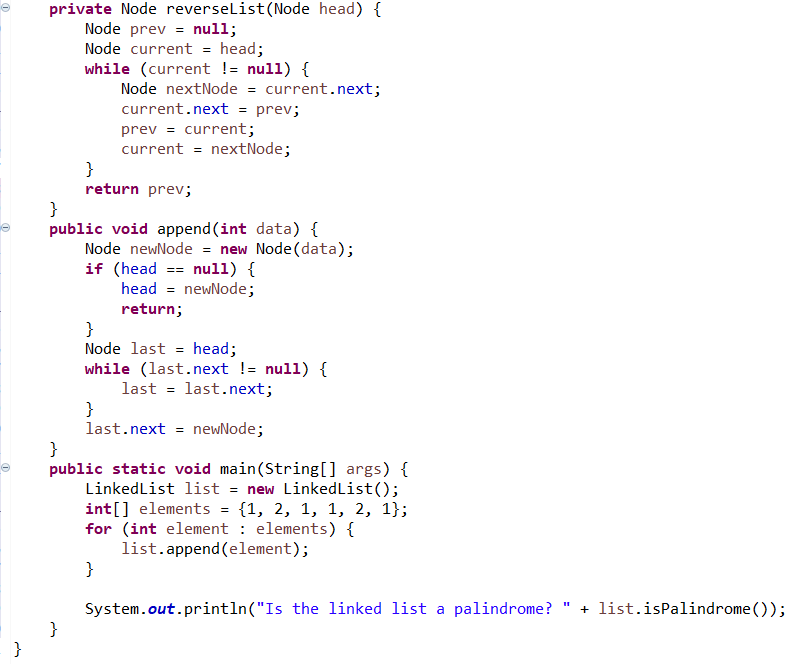
1. **Balanced Tree Check:**

Given a binary tree, find if it is height balanced or not. A tree is height balanced if difference between heights of left and right subtrees is not more than one for all nodes of tree.

**Time Complexity: O(n)**

**Code:**

****

****

**Input:**

1

/

2

\

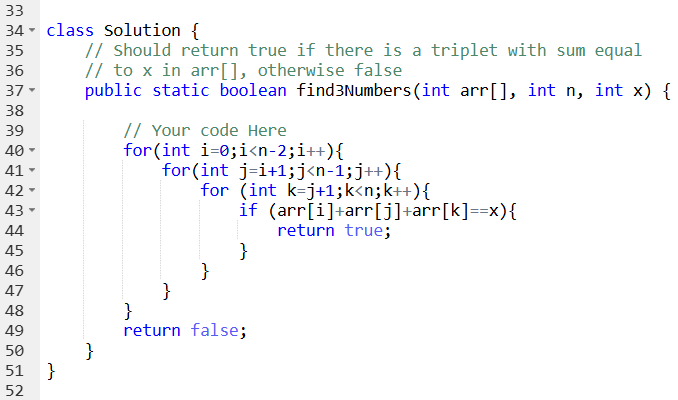
3

**Output:** 0

1. **Triplets Sum in Array:**

Given an array arr of size n and an integer x. Find if there's a triplet in the array which sums up to the given integer x. Examples Input:n = 6, x = 13, arr[] = [1,4,45,6,10,8] Output: 1 Explanation: The triplet {1, 4, 8} in the array sums up to 13.

**Time Complexity: O(n^3)**



**Input**:n = 6, x = 13, arr[] = [1,4,45,6,10,8]

**Output**: 1