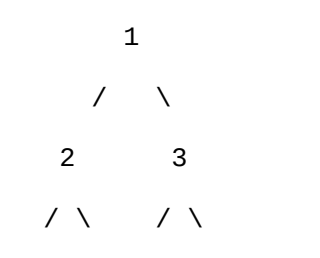
Assignment Questions 20

Question-1

Given a binary tree, your task is to find subtree with maximum sum in tree.

Examples:

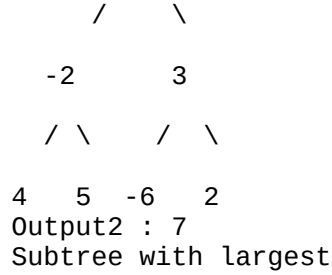
Input1 :



Output1 : 28

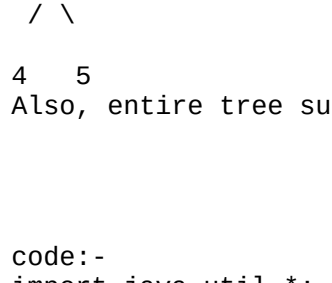
As all the tree elements are positive, the largest subtree sum is equal to sum of all tree elements.

Input2 :



Output2 : 7

Subtree with largest sum is :



Also, entire tree sum is also 7.

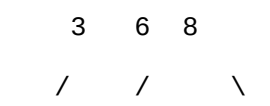
code:-

```
import java.util.*;
class GFG
{
static class Node
{
      int key;
      Node left, right;
}

static class INT
{
      int v;
      INT(int a)
      {
```

```
            v = a;
      }
}
static Node newNode(int key)
{
      Node temp = new Node();
      temp.key = key;
      temp.left = temp.right = null;
      return temp;
}
static int findLargestSubtreeSumUtil(Node root,INT ans)
{
      if (root == null)
            return 0;
      int currSum = root.key +
      findLargestSubtreeSumUtil(root.left, ans) +
      findLargestSubtreeSumUtil(root.right, ans);
      ans.v = Math.max(ans.v, currSum);
      return currSum;
}
static int findLargestSubtreeSum(Node root)
{
      if (root == null)
            return 0;
      INT ans = new INT(-9999999);
      findLargestSubtreeSumUtil(root, ans);
      return ans.v;
}

public static void main(String args[])
{
      Node root = newNode(1);
      root.left = newNode(-2);
      root.right = newNode(3);
      root.left.left = newNode(4);
      root.left.right = newNode(5);
      root.right.left = newNode(-6);
      root.right.right = newNode(2);

      System.out.println(findLargestSubtreeSum(root));
}
}
```

Question-2
Construct the BST (Binary Search Tree) from its given level order traversal.
Example:
Input: arr[] = {7, 4, 12, 3, 6, 8, 1, 5, 10}
Output: BST:

```
7

         /    \

       4     12

     /  \     /
```

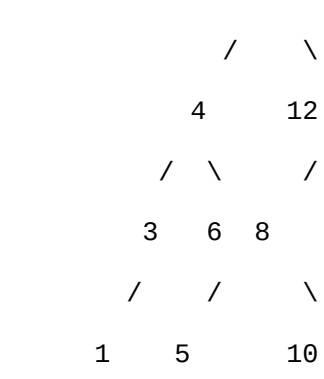


coode:-

```
import java.io.*;
class GFG {
	static class Node {
		int data;
		Node left, right;
	};
	static Node getNode(int data)
	{
		Node newNode = new Node();
		newNode.data = data;
		newNode.left = newNode.right = null;
		return newNode;
	}
	static Node LevelOrder(Node root, int data)
	{
		if (root == null) {
			root = getNode(data);
			return root;
		}
		if (data <= root.data)
			root.left = LevelOrder(root.left, data);
		else
			root.right = LevelOrder(root.right, data);
		return root;
	}

	static Node constructBst(int arr[], int n)
	{
		if (n == 0)
			return null;
		Node root = null;

		for (int i = 0; i < n; i++)
			root = LevelOrder(root, arr[i]);

		return root;
	}
	static void inorderTraversal(Node root)
	{
		if (root == null)
			return;

		inorderTraversal(root.left);
		System.out.print(root.data + " ");
		inorderTraversal(root.right);
	}
	public static void main(String args[])
	{
		int arr[] = { 7, 4, 12, 3, 6, 8, 1, 5, 10 };
		int n = arr.length;
```

```
            Node root = constructBst(arr, n);
            System.out.print("Inorder Traversal: ");
            inorderTraversal(root);
      }
}
```

Question-3
Given an array of size n. The problem is to check whether the given array can represent the level order traversal of a Binary Search Tree or not.
Examples:
Input1 : arr[] = {7, 4, 12, 3, 6, 8, 1, 5, 10}
Output1 : Yes
For the given arr[], the Binary Search Tree is:



Input2 : arr[] = {11, 6, 13, 5, 12, 10}
Output2 : No
The given arr[] does not represent the level order traversal of a BST.

code:-

```
import java.util.*;
public class Solution
{
static class NodeDetails
{
      int data;
      int min, max;
};

static boolean levelOrderIsOfBST(int arr[], int n)
{
      if (n == 0)
            return true;
      Queue<NodeDetails> q = new LinkedList<NodeDetails>();
      int i = 0;
      NodeDetails newNode=new NodeDetails();
      newNode.data = arr[i++];
      newNode.min = Integer.MIN_VALUE;
      newNode.max = Integer.MAX_VALUE;
      q.add(newNode);
      while (i != n && q.size() > 0)
      {
            NodeDetails temp = q.peek();
            q.remove();
            newNode = new NodeDetails();
```

```
            if (i < n && (arr[i] < (int)temp.data && arr[i] > (int)temp.min))
            {
                  newNode.data = arr[i++];
                  newNode.min = temp.min;
                  newNode.max = temp.data;
                  q.add(newNode);
            }

            newNode=new NodeDetails();
            if (i < n && (arr[i] > (int)temp.data && arr[i] < (int)temp.max))
            {
                  newNode.data = arr[i++];
                  newNode.min = temp.data;
                  newNode.max = temp.max;
                  q.add(newNode);
            }
      }
      if (i == n)
            return true;
      return false;
}
public static void main(String args[])
{
      int arr[] = {7, 4, 12, 3, 6, 8, 1, 5, 10};
      int n = arr.length;
      if (levelOrderIsOfBST(arr, n))
            System.out.print( "Yes");
      else
            System.out.print( "No");

}
}
```