**package** pair\_withgiven\_sum.java;

// program to find a pair with given sum

// in a Balanced BST

**import** java.util.ArrayList;

**import** java.util.Scanner;

// A binary tree node

**class** Node1 {

**int** data1;

Node1 left1, right1;

Node1(**int** d){

data1 = d;

left1 = right1 = **null**;

}

}

**public** **class** BinarySearchTree {

//root of the BST

Node1 root1;

// constructor

BinarySearchTree(){

root1 = **null**;

}

// inorder traversal of the tree

**void** inorder(){

inorderUtil1(**this**.root1);

}

// utility function for inorder traversal of tree

**void** inorderUtil1(Node1 node1){

**if** (node1 == **null**)

**return**;

inorderUtil1(node1.left1);

System.***out***.print(node1.data1 + " ");

inorderUtil1(node1.right1);

}

// this method calls insertRec()

**void** insert(**int** key1){

root1 = insertRec1(root1, key1);

}

//recursive function to insert a new key in BST

Node1 insertRec1(Node1 root1, **int** data1){

//if tree is empty, return new node

**if** (root1 == **null**) {

root1 = **new** Node1(data1);

**return** root1;

}

//Otherwise, recur down the binary tree

**if** (data1 < root1.data1)

root1.left1 = insertRec1(root1.left1, data1);

**else** **if** (data1 > root1.data1)

root1.right1 = insertRec1(root1.right1, data1);

**return** root1;

}

// method that will add values of given BST into the ArrayList

// and returns the ArrayList

ArrayList<Integer> treeToList(Node1 node1, ArrayList<Integer> list1){

// Base case

**if** (node1 == **null**)

**return** list1;

treeToList(node1.left1, list1);

list1.add(node1.data1);

treeToList(node1.right1, list1);

**return** list1;

}

// Indicates method which checks if there is a pair present

**public** **static** **boolean** isPairPresent(Node1 node1, **int** target1){

// Now this list a1 is passed as an argument

// in treeToList method

// which is later on filled by the values of BST

ArrayList<Integer> a1 = **new** ArrayList<>();

// Now a2 list contains all the values of BST

// returned by treeToList method

ArrayList<Integer> a2 = treeToList(node1, a1);

**int** start1 = 0; // Indicates starting index of a2

**int** end1 = a2.size() - 1; // Indicates ending index of a2

**while** (start1 < end1) {

**if** (a2.get(start1) + a2.get(end1) == target1)

// Target Found!

{

System.***out***.println("Pair Found: " + a2.get(start1) + " + " + a2.get(end1) + " " + "= " + target1);

}

**return** **true**;

}

**return** **false**;

}

// Driver function

**public** **static** **void** main(String[] args){

BinarySearchTree tree1 = **new** BinarySearchTree();

/\*

40

/ \

20 60

/ \ / \

10 30 50 70 \*/

tree1.insert(40);

tree1.insert(20);

tree1.insert(10);

tree1.insert(30);

tree1.insert(50);

tree1.insert(60);

tree1.insert(70);

Scanner sc = **new** Scanner(System.***in***);

**int** start1= sc.nextInt();

System.out.println("Enter desired sum " + target1);

**if**(isPairPresent(**true**)) {

System.out.println("Pair Found: " + a2.get(start1) + " + "+ a2.get(end1) + " " + "= " + target1)

}

**else** {

System.out.println("Nodes are not found");

}

}

}