```java
package binarytree;

//Java implementation to check if given Binary tree
//is a BST or not

/* Class containing left and right child of current
node and key value*/


 class BinaryTree33
{
    //Root of the Binary Tree
    Node root;

    /* can give min and max value according to your code or
    can write a function to find min and max value of tree. */

    /* returns true if given search tree is binary
    search tree (efficient version) */
    boolean isBST() {
        return isBSTUtil(root, Integer.MIN_VALUE,
                Integer.MAX_VALUE);
    }

    /* Returns true if the given tree is a BST and its
    values are >= min and <= max. */
    boolean isBSTUtil(Node node, int min, int max)
    {
        /* an empty tree is BST */
        if (node == null)
            return true;

        /* false if this node violates the min/max constraints */
        if (node.data < min || node.data > max)
            return false;

        /* otherwise check the subtrees recursively
        tightening the min/max constraints */
        // Allow only distinct values
        return (isBSTUtil(node.left, min, node.data-1) &&
                isBSTUtil(node.right, node.data+1, max));
    }

    /* Driver program to test above functions */
    public static void main(String args[])
    {
        BinaryTree33 tree = new BinaryTree33();
        tree.root = new Node(4);
        tree.root.left = new Node(2);
        tree.root.right = new Node(5);
        tree.root.left.left = new Node(1);
        tree.root.left.right = new Node(3);

        if (tree.isBST())
            System.out.println("IS BST");
        else
            System.out.println("Not a BST");
    }

}
```