Checking BST

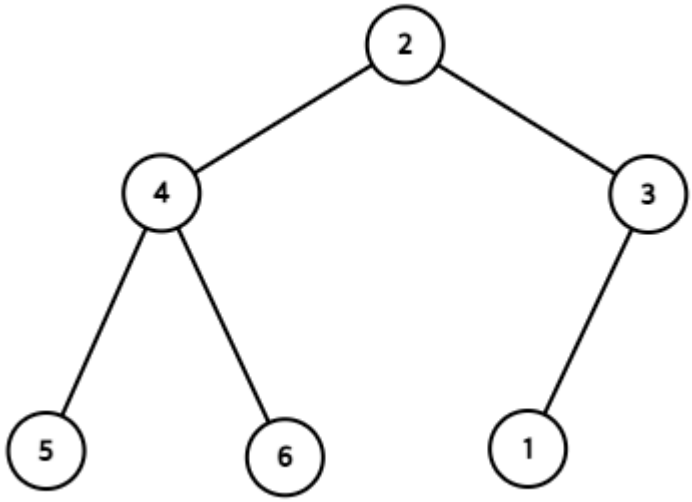
We define a binary tree to be a binary search tree with the following ordering requirements:

- The value of every node in a node's left subtree is *less than* the data value of that node.
- The value of every node in a node's right subtree is *greater than* the data value of that node.

Given the root node of a binary tree, can you determine if it's also a binary search tree?

Create a program that uses node structure to connect a given tree and make `is_BST()` function that receives tree’s root node as a parameter to print out whether it is BST or not.

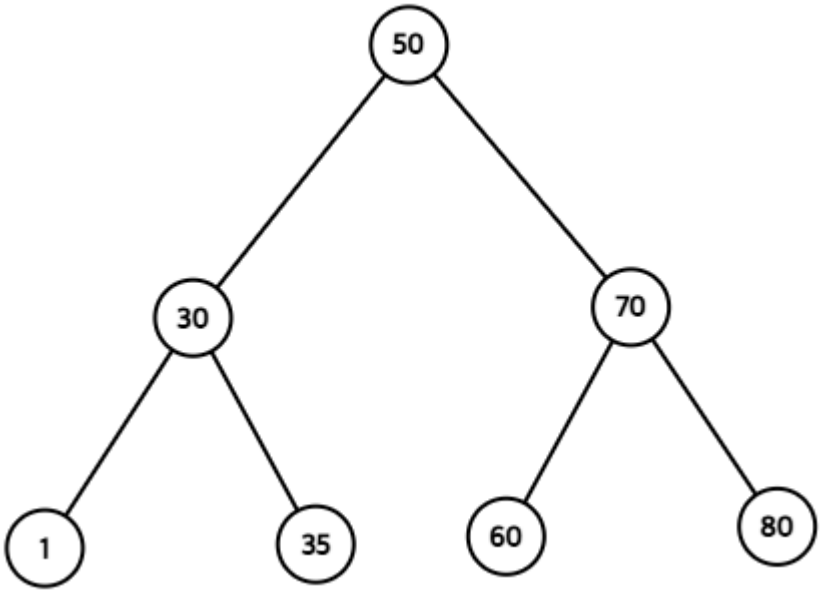
Input 1



Output 1

“This is not BST”

Input 2



Output 2

“This is BST”

Answer Code Example

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

typedef struct Node
{
    int key;
    struct Node *left;
    struct Node *right;
} Node;

int arr[1000];
int n = 0;

void inorder_traversal(Node *root)
{
    if (root == NULL) return;

    inorder_traversal(root->left);
    arr[n++] = root->key; /* store route of traversal */
    inorder_traversal(root->right);
}

bool is_BST(Node *root)
{
    inorder_traversal(root);

    for (int i = 0; i < n-1; i++)
    {
        if (arr[i] >= arr[i+1])
            return false;
    }
    return true;
}

int main()
{
    bool result = false;
    /* input 1 */
    // Node* node1 = (Node*)malloc(sizeof(Node));
    // node1->key = 2;

    // Node* node2 = (Node*)malloc(sizeof(Node));
    // node2->key = 4;

    // Node* node3 = (Node*)malloc(sizeof(Node));
    // node3->key = 3;

    // Node* node4 = (Node*)malloc(sizeof(Node));
```

```
    // node4->key = 5;

    // Node* node5 = (Node*)malloc(sizeof(Node));
    // node5->key = 6;

    // Node* node6 = (Node*)malloc(sizeof(Node));
    // node6->key = 1;

    // node1->left = node2;
    // node1->right = node3;
    // node2->left = node4;
    // node2-> right = node5;
    // node3->left = node6;

    /* input 2 */
    // Node* node1 = (Node*)malloc(sizeof(Node));
    // node1->key = 50;

    // Node* node2 = (Node*)malloc(sizeof(Node));
    // node2->key = 30;

    // Node* node3 = (Node*)malloc(sizeof(Node));
    // node3->key = 70;

    // Node* node4 = (Node*)malloc(sizeof(Node));
    // node4->key = 1;

    // Node* node5 = (Node*)malloc(sizeof(Node));
    // node5->key = 35;

    // Node* node6 = (Node*)malloc(sizeof(Node));
    // node6->key = 60;

    // Node* node7 = (Node*)malloc(sizeof(Node));
    // node7->key = 80;

    // node1->left = node2;
    // node1->right = node3;
    // node2->left = node4;
    // node2->right = node5;
    // node3->left = node6;
    // node3->right = node7;

    result = is_BST(node1);
    if (result == true)
    {
        printf("This is BST");
    } else if (result == false)
    {
        printf("This is not BST");
    }
}
```