Check if a Tree is Binary Search Tree

```
#include <stdio.h>
#include <stdlib.h>
 struct node
{
   int data;
   struct node* left;
   struct node* right;
};
 static struct node *prev = NULL;
 /*Function to check whether the tree is BST or not*/
int is_bst(struct node* root)
{
   //Write your code here
}
struct node* newNode(int data)
{
   struct node* node = (struct node*)malloc(sizeof(struct node));
   node->data = data;
   node->left = NULL;
   node->right = NULL;
    return(node);
}
 int main()
{
  /*
   The input tree is as shown below

            40

            / \

         20      60
```

```
        / \     \

      10      30     80

                \

                  90

 */
  struct node *root = newNode(40);
  root->left      = newNode(20);
  root->right     = newNode(60);
  root->left->left  = newNode(10);
  root->left->right = newNode(30);
  root->right->right = newNode(80);
  root->right->right->right = newNode(90);
  if (is_bst(root))
    printf("TREE 1 Is BST");
  else
    printf("TREE 1 Not a BST");
  prev = NULL;
/*
  The input tree is as shown below
          50
          / \
        20      30
        / \

     70      80

     / \       \

  10    40    60
*/
```

```
    struct node *root1 = newNode(50);
    root1->left  = newNode(20);
    root1->right  = newNode(30);
    root1->left->left  = newNode(70);
    root1->left->right = newNode(80);
    root1->left->left->right = newNode(40);
    root1->left->left->left= newNode(90);
    if (is_bst(root1))
       printf("TREE 2 Is BST");
    else
       printf("TREE 2 Not a BST");
    return 0;
}
```