

Java 1996 AP Computer Science Question 3

Assume that binary trees are implemented using the standard `TreeNode` class:

```
public class TreeNode
{
    public TreeNode(Object initValue, TreeNode initLeft, TreeNode initRight)
    {
        // not shown
    }

    public Object getValue() { // not shown }
    public TreeNode getLeft() { // not shown }
    public TreeNode getRight() { // not shown }

    public void setValue(Object theNewValue) { // not shown }
    public void setLeft(TreeNode theNewLeft) { // not shown }
    public void setRight(TreeNode theNewRight) { // not shown }
};
```

Part A

Write the method `valsLess` whose header is given below. `ValsLess` returns *true* if its parameter *t* is null or if all values stored in the binary tree pointed to by *t* are less than *obj*; otherwise `valsLess` returns *false*. Assume all values stored in *t* implement the `Comparable` interface.

Complete `ValsLess` below the following header.

```
// precondition: all values stored in t implement Comparable
// postcondition: returns true if t is null or if
//                all values stored in tree represented
//                by t are less than obj; otherwise
//                returns false

public boolean valsLess(TreeNode t, Comparable obj)
```

[answer](#)

Part B

Recall that a binary tree *T* that contains no duplicate values is a search tree if and only if

1. *T* is empty or
2. all of the following are true
 - The value stored at the root of *T* is greater than all of the values stored in *T*'s left subtree.
 - The value stored at the root of *T* is less than all of the values stored in *T*'s right subtree.
 - *T*'s left subtree is a binary search tree that contains no duplicate values.
 - *T*'s right subtree is a binary search tree that contains no duplicate values.

Write function `isBST` whose header is given below. `isBST` should return *true* if the binary tree represented by its parameter *t* is a binary search tree that contains no duplicate values; otherwise `isBST` should return *false*.

In writing `isBST` you may include calls to function `valsLess` from part (A). Assume that `valsLess` works as specified, regardless of what you wrote in part (A). You may also include calls to function `valsGreater` whose specification is given below. (You do not need to write the body of `valsGreater`.)

```
// precondition: all values stored in t implement Comparable
```

```
// precondition: returns true if t is null or if
//               all values stored in tree represented
//               by t are greater than obj; otherwise
//               returns false

public boolean valsGreater(TreeNode t, Comparable obj)
```

Complete function *isBST* below the following header.

```
// precondition: returns true if t represents a binary search
//               tree containing no duplicate values;
//               otherwise, returns false.

public boolean isBST(TreeNode t)
```

[answer](#)

[Owen L. Astrachan](#)

Last modified: Sat Jul 5 10:45:01 EDT 2003