Homework #5 - Binary Search Trees

Note: Follow my naming exactly!

Write a binary tree class to hold any one data type using generics.

Function signatures / Variables - Swift:

BinarySearchTree<T : Comparable>
var height : Int
var isEmpty : Bool
var size : Int
var elements : [T] // in order
init () // make an empty tree
func insert(element : T)
func contains(element : T) -> Bool
func search(element : T) -> T? //return the stored element if you find it, nil if you don't
func makeBreadthFirstArray() -> [T] //Top to bottom, left to right
Optional
init (fromSortedData : [T])
func delete(element : T)

Function signatures / Variables - C++:

BinarySearchTree
BinarySearchTree() // makes an empty one
int getHeight() const
bool isEmpty() const
int getSize() const
std::vector<T> elementVector() const
void insert(const T&)
bool contains(const T&) const

BinarySearchTree
T search(const T&) const
std::vector<T> makeBreadthFirstVector() const //Top to bottom, left to right
Optional
BinarySearchTree(const std::vector<T>&) //builds tree from sorted array of ints
void delete(const T&)

Written

1) Draw a tree for which the preorder and inorder traversals generate the same sequence
2) Design an algorithm (pseudocode or prose) to check if a binary tree is a binary search tree
3) Design an algorithm (pseudocode or prose) to check if a binary tree is perfectly balanced (smallest height possible for the number of nodes)