Do not use `Struct` in any of your Tree Lab sheets, and for any Tree Traversals. Always code a private recursive version and make sure your public versions do not take arguments. For non-templated tasks make sure to separate your implementation code (.cpp) from declaration code (.h)

Part A

1. Code a TreeNode Class which contains two TreeNode pointers and an integer data member.

2. Code a Binary Search Tree class for integers, implement the following public interface:

   ```
   BinarySearchTree(); // constructor
   void add(int toadd); // both the add and height methods are
   int height() // implemented by a private recursive method
   ```

3. Code a main program which adds several integers to a binary tree and prints the height of the tree.

   Note: You will have two add and two height methods, one public and one private. So for example, you need to provide an extra private method `add(TreeNode *toAdd, TreeNode *addHere)` to call recursively.

   Note: For a tree with just one node, the root node, the height is defined to be 0, if there are 2 levels of nodes the height is 1 and so on. A null tree (no nodes except the null node) is defined to have a height of -1.

4. Code a remove method `void remove(int toremove);` and the corosponding private method to remove a node containing the `toremove` integer value from the tree. If the value is not in the Tree print a message.

5. Code a `~BinarySearchTree(); // deconstructor`

Part B

6. Starting a new Visual Studio solution, write the code for a binary search tree of chars, and use a recursive method to search the tree for a particular character and return true if it is present, false otherwise.

7. Add the code to print all the values in the char binary search tree in ascending order.

8. Add code to print all values in Pre Order.

9. Add code to print all values in Post Order.