You will submit your solution to this assignment to the Curator System (as HW1). Your solution must be either a plain text file (e.g., NotePad) or a document that can be edited in MS Word; submissions in other formats will not be graded.

Partial credit will only be given if you show relevant work.

1. [40 points] Design an algorithm to determine whether a given binary tree is organized to support the BST property. Express your solution as a Java function (not a BST member function) implemented in the same package as the BST generic specified in Minor Project 1:

   ```java
   boolean isValidBST( BST<T> Tree ) {
     if ( Tree.root == null )
       return true;

     return isValidBSTHelper(Tree.root, null, null);
   }

   boolean isValidBSTHelper(BinaryNode sRoot, T floor, T ceiling) {

     // empty tree is valid
     if ( sRoot == null ) return true;

     // see if current element is smaller than floor value
     if ( floor != null && sRoot.element.compareTo(floor) == -1 )
       return false;

     // see if current element is larger than ceiling value
     if ( ceiling != null && sRoot.element.compareTo(ceiling) == 1)
       return false;

     // check both subtrees
     // values in left subtree must conform to floor for this node
     // and have value in this node as their ceiling
     // values in right subtree have value in this node as their
     // floor, and must conform to ceiling for this node
     return isValidBSTHelper(sRoot.left, floor, sRoot.element) &&
        isValidBSTHelper(sRoot.right, sRoot.element, ceiling);
   }
   ```
2. [30 points] Use Induction to prove the following fact: for every integer \( \lambda \geq 1 \), a BST with \( \lambda \) levels can contain no more than \( 2^\lambda - 1 \) nodes.

Proof: Suppose that \( \lambda = 1 \), then the tree consists of a single level, which must contain the root node and nothing else; therefore, the number of nodes is 1, and that equals \( 2^1 - 1 \).

Now, suppose that for some value \( k \geq 1 \), every BST that has \( \lambda \) levels, where \( \lambda \leq k \), has no more than \( 2^\lambda - 1 \) nodes.

Suppose that we have a BST with \( k + 1 \) levels. Then it must have at least 2 levels, and hence it has a root node with at least one non-empty subtree. Call the subtrees of the root node L and R. Then, since L has no more than \( k \) levels, by the inductive assumption above, L has no more than \( 2^k - 1 \) nodes. By the same logic, R has no more than \( 2^k - 1 \) nodes.

So, the total number of nodes in the tree cannot be more than \( 1 + (2^k - 1) + (2^k - 1) \), allowing for the root node. But the last expression reduces to \( 2 \cdot 2^k - 1 \) or \( 2^{k+1} - 1 \).

Therefore, any BST that has \( k + 1 \) levels cannot have more than \( 2^{k+1} - 1 \) nodes.

Therefore by the (strong) Principle of Induction, the theorem is proved.

3. [30 points] An algorithm takes 0.5 ms to execute on an input of size 100. Assuming that the low-order terms are insignificant, how long will the same algorithm take to execute on an input of size 1000 if its complexity is:

The key here is to understand what the complexity result means about the way the number of operations, and hence the time, changes as the size of the input increases. If you think of having a function \( T(N) \) that gives the number of operations needed for an input of size \( N \), and \( T(N) \) is \( O(f(N)) \), then if we disregard small terms we know that \( T(N) = kf(N) \) for some constant \( k \).

a) \( O(N) \)

In this case, \( T(N) = kN \), and so \( \frac{T(1000)}{T(100)} \approx \frac{1000k}{100k} \approx 10 \), so an input of size 1000 would take about 10 times longer, or about 5 ms.

b) \( O(N \log N) \)

In this case, \( T(N) = kN \log N \), and so \( \frac{T(1000)}{T(100)} \approx \frac{k1000 \log 1000}{k100 \log 100} \approx 10 \frac{\log 1000}{\log 100} \approx 14 \), so an input of size 1000 would take about 14 times longer, or about 7 ms.

c) \( O(N^2) \)

In this case, \( T(N) = kN^2 \), and so \( \frac{T(1000)}{T(100)} \approx \frac{k1000^2}{k100^2} \approx 100 \), so an input of size 1000 would take about 100 times longer, or about 50 ms.