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 typed MS Word document; submissions in other formats will not be graded.

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

---

1. [25 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 2:

```
boolean isValidBST( BST<T> Tree ) {

   if ( Tree == null ) return false;
   if ( Tree.isEmpty() ) return true;

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

boolean isValidBST(BinaryNode<T> sRoot, T Lo, T Hi) {

   if ( sRoot == null ) return true;

   if ( Lo != null && sRoot.element.compareTo(Lo) <= 0 )
      return false;

   if ( Hi != null && sRoot.element.compareTo(Hi) >= 0 )
      return false;

   return isValidBST(sRoot.left, Lo, sRoot.element) &&
          isValidBST(sRoot.right, sRoot.element, Hi);
}
```

2. [25 points] Write an implementation of an algorithm to perform a range search in a BST. Base your solution on the BST interface given for Minor Project 1, and

Assume that the following public method has been added to the interface for the BST given in Minor Project 1:

```
// Pre:      lower and upper are valid objects of type T, such that
//           lower <= upper, according to type T's compareTo()
// Returns:  Vector object containing all the elements X found in the
//           BST such that lower < X < upper, according to compareTo();
//           the order in which the elements occur is not guaranteed
//
public Vector<T> rangeSearch(T lower, T upper) {

   Vector<T> matches = new Vector<T>();

   rangeSearchHelper(lower, upper, root, matches);

   return matches;
}
```

Complete the implementation of the following private helper function, which would also be added to the given BST interface:

```
private void rangeSearchHelper(T lower, T upper, BinaryNode sroot,
                                                    Vector<T> matches) {

    if ( sroot == null ) return;   // nothing of interest here

    if ( sroot.element.compareTo(lower) > 0 &&
         sroot.element.compareTo(upper) < 0 ) { // current elem is in range
        matches.add(sroot.element);
    }
    if ( sroot.element.compareTo(lower) > 0 ) {  // may be matches in left
                                                 //   subtree
       rangeSearchHelper(lower, upper, sroot.left, matches);
    }
    if ( sroot.element.compareTo(upper) <= 0 ) {  // may be matches in right
                                                  //   subtree
       rangeSearchHelper(lower, upper, sroot.right, matches);
    }
}
```

Your implementation should operate as efficiently as possible. It should put references to all the matching data objects, if any, into the `Vector` object that is returned by the public function.

3. [25 points] Use Induction to prove the following fact: for every integer, $N \geq 1$, a BST with N nodes must have at least $\lceil \log(N+1) \rceil$ levels. (You may not use any of the BST theorems from the notes.)

If T is a BST with *N* = 1 node then T has 0 levels, and $\lceil \log(N+1) \rceil = \lceil \log(1+1) \rceil = \lceil \log(2) \rceil = 1$.

If Tis a BST with 2 or 3 nodes, then T has at least 2 levels, and $\lceil \log(N+1) \rceil \leq \lceil \log(3+1) \rceil = \lceil \log(4) \rceil = 2$.

Now, suppose that for some *N*, if T is a BST with *K* nodes, where $0 \leq K \leq N$, then T the number of levels in T is at least $\lceil \log(K+1) \rceil$.

Let T be a BST with *N* + 1 nodes. Then T consists of a root node and two subtrees, and the two subtrees must collectively contain *N* nodes. Therefore, one of the subtrees must contain at least *N*/2 nodes.

Then, by the inductive assumption, the number of levels in that subtree must be at least $\lceil \log(N/2+1) \rceil$.

And so, T must contain at least $\lceil \log(N/2+1) \rceil + 1$ levels. However:

$$\begin{aligned} \lceil \log(N/2+1) \rceil + 1 &= \lceil \log(N/2+1) + 1 \rceil \\ &= \lceil \log(N/2+1) + \log(2) \rceil \\ &= \lceil \log(2(N/2+1)) \rceil \\ &= \lceil \log(N+2) \rceil \\ &= \lceil \log((N+1)+1) \rceil \end{aligned}$$

Therefore, by induction, the theorem holds for all $N \geq 1$.

4. [25 points] Use the result proved in question 3 to prove that: for every integer, $\lambda \geq 1$, a BST with λ levels can contain no more than $2^{\lambda} - 1$ nodes.

Suppose that T is a BST with at least 2^{λ} nodes. Then, from problem 3, the number of levels in T must be at least $\lceil \log(2^{\lambda}+1) \rceil$.

Now, log() is strictly increasing, so $\lceil \log(2^{\lambda}+1) \rceil > \lceil \log(2^{\lambda}) \rceil = \lambda$, and therefore (since λ is an integer) $\lceil \log(2^{\lambda}+1) \rceil \geq \lambda + 1$.

But, that contradicts the assumption that T has λ levels.