Prepare your answers to the following questions in a Word document or a plain text file. Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted.

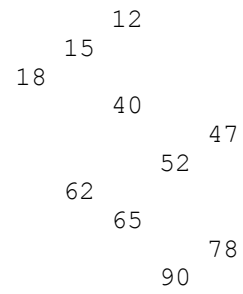
You will submit your answers to the Curator System (www.cs.vt.edu/curator) under the heading HW3.

For questions 1 and 2, assume you are working with the node and tree templates specified for Minor Project 2.

1. [15 pts] Design and implement a private helper member function for the `BST` template that conforms to the following interface:

```
// extractLeftChildren() returns a vector object containing copies of
// all the data elements in the tree that occur in nodes that are left
// children of other nodes.  The tree is not modified.
//
// Parameters:  none
// Pre:         none
// Post:        specified value is returned, tree is unchanged
// Called by:   client
// Calls:       eLCHelper()
template <typename T> vector<T> BST<T>::extractLeftChildren() const {

      vector<T> Lefties;
      eLCHelper(Root, Lefties);
      return Lefties;
}

// eLCHelper() traverses the tree and creates and returns a vector
// object containing copies of all data elements that occur in left
// children within the tree.
//
// Parameters:  sRoot    pointer to a tree node, or NULL
//              Lefties  vector in which values will be placed
// Pre:         sRoot is NULL or targets a tree node, Lefties is
//              empty on initial (non-recursive) call
// Post:        Lefties contains the specified values, tree is unchanged
// Called by:   extractLeftChildren()
// Calls:       itself, std::vector interface, possibly other member fns
template <typename T>
void BST<T>::eLCHelper(BinNodeT<T>* sRoot, vector<T>& Lefties) const {

   // . . .

}
```

For example, given the following binary tree:



a call to `extractLeftChildren()` would return a vector containing the values 12, 15, 40, 47 and 78. The order in which the elements are stored in the vector is unspecified and will depend upon the traversal pattern used in your solution.

2. [15 pts] Write a recursive client function (non-member, non-friend) which will take a sorted list of integers and insert them into a `BST` object so that the resulting tree has the minimum possible height for the given number of elements.

Your solution must conform to the following interface:

```
// mapSortedListToBST() inserts the elements of the array List[] into
// the binary search tree Tree so that Tree has minimal height.
//
// Parameters:  Lo     index of first elem of List[] to be processed
//              Hi     index of final elem of List[] to be processed
//              List[] array of values to place into tree
//              Tree   BST object to receive values
// Pre:         Lo and Hi are indices of data elements in List[]
//              List[i] <= List[i+1] for 0 <= i < Hi
//              for initial (non-recursive) call, Tree is empty
// Post:        List[] is unchanged
//              Tree is a minimal-height binary search tree containing
//              the values between Lo and Hi, inclusive
// Called by:   client
// Calls:       BST<T>::Insert()
void mapSortedListToBST(int Lo, int Hi, int List[], BST<int>& Tree) {
   // . . .
}
```

For example, given an array containing the values:

73, 81, 85, 86, 94, 98, 105, 115, 121, 124, 133, 138, 146, 154, 155, 163, 168, 173, 174, 182

then a call to `mapSortedListToBST()` might result in the following binary search tree:

```
          73
      81
          85
             86
   94
          98
      105
          115
             121
124
          133
      138
          146
             154
   155
          163
             168
      173
          174
             182
```

3. [10 pts] Use mathematical induction to prove the following theorem about binary trees. "Prove" means provide a formal, mathematical argument as you would have done in Math 2534. It does not mean provide a disconnected list of mathematical formulas with little or no accompanying explanation.

Theorem If T is a binary tree with λ levels then the number of nodes N in T satisfies the bound: $N \leq 2^{\lambda} - 1$.