CSCI 340 **Computer Assignment 6** **Fall 2017**
(10 points)

Binary Tree Class

For this computer assignment, you are to write a C++ program to implement a class for binary trees. To deal with variety of data types, implement this class as a template.

The definition of the class for a binary tree (as a template) is given as follows:

```
template < class T > class binTree {
public:
    binTree ( );                                        // default constructor
    unsigned height ( ) const;                          // returns height of tree
    virtual void insert ( const T& );                   // inserts a node in tree
    void inorder ( void ( * ) ( const T& ) );           // inorder traversal of tree
protected:
    Node < T >* root;                                   // root of tree
private:
    unsigned height ( Node < T >* ) const;              // private version of height ( )
    void insert ( Node < T >*&, const T& );             // private version of insert ( )
    void inorder ( Node < T >*, void ( * ) ( const T& ) ); // private version of inorder ( )
};
```

Most of the *public* member functions of the binTree class call *private* member functions of the class (with the same name). These *private* member functions can be implemented as either *recursive* or *non-recursive,* but clearly, *recursive* versions of these functions are preferable because of their short and simple implementations in code. However, they require more memory and usually slower than their *non-recursive* versions in execution, especially for a large amount of input data.

Because of information hiding, a client is not permitted to access the binary tree directly, so the *root* of the tree is kept *protected* (not *private* because of future implementations of *derived* classes from the *base* class of the binTree), so it cannot be passed as an argument to any of the *public* functions of the tree. It is essential to have *private* utility functions, which act as interface between a client and the tree. The insert ( ) function of the binTree class is described as follows:

insert (const T& x) : This *virtual* function can be used to insert a node with the data value x in a binary tree, applying the following technique: if the tree is empty, then the new node will be the *root* of the tree with the value x; otherwise, the left or the right subtree is randomly selected and the value x is inserted in that side. To implement the random selection, you can use the following RNG.

```
typedef enum { left_side, right_side } SIDE;
SIDE rnd ( ) { return rand ( ) % 2 ? right_side : left_side; }
```

Put the implementation of your binTree class in the following header file:

- binTree.h: Definition of the class Node, which represents the nodes in a binary tree, can be found in the header file Node.h. To use the class Node in your program, include the header file Node.h by inserting the statement: #include "/home/cs340/progs/17f/p6/Node.h" at the top of your header file.

The source file prog6.cc contains the driver program. In addition to the main ( ) routine, it has the implementations of the following routines (as templates) and the definitions of the two RNGs used in the main ( ) routine.

- template < class T > void print (const T& x):
- template < class T > void print_vals (binTree < T >& tree, const string& name);

The unary function print ( ) can be used as an argument to the member functions inorder ( ) to print the value of its argument x. The function print_vals ( ) does the followings: first, it prints name, which is the name of the tree, and it also prints the height of the tree. Then, it calls the member function inorder ( ) to print the data values in the tree in inorder. The class RND1 can be used to generate random integers in the range [LOW1 = –999, HIGH1 = 999] and the class RND2 can be used to generate random floating-point numbers in the range [LOW2 = –999.99, HIGH2 = 999.99]. The function objects RND1 ( ) and RND2 ( ), generated from these classes, are used to fill in the random values in vector containers vector < int > A (N1) and vector < float > B (N2) by using the generate ( ) function in STL, where N1 = 100 and N2 = 50 are the corresponding sizes of these two vectors. The main ( ) routine copies the random values from vectors A and B and inserts them in the binary trees first and second, respectively. At the end, the data values in the binary trees first and second are printed out on stdout with LSIZE = 12 numbers in a single line.

The source file of the driver program prog6.cc, in directory: ~cs340/progs/17f/p6, is provided. To compile and link the driver program with the system library routines, execute: Make N=6. To test your binary-tree class, execute: Make execute N=6. This command executes the driver program and displays the output as well as any error messages both on the terminal screen and in prog6.out. After you are done with the program, you don't need its object and executable files any more. To delete them, execute: Make clean.

The correct output of this program can be found in file prog6.out, which is in the same directory with prog6.cc. If the program fails to generate the correct output, then to debug the program, copy the file prog6.cc in your own directory, uncomment some of the statements in that file, recompile and link the modified program, and then execute again.

When your program is ready, submit the header file of your binary-tree class to your TA, executing: mail_prog.340 binTree.h.