

## CSSE 250 - Assignments #4 - Spring 2009

Due by 8:15Am, Tuesday 5/12/2009

Implement the BST template class and write a client program to test it. The implementations of member functions must follow the declarations provided below. Submission instructions will be posted on the website.

```
template <typename T>
class BST {
public:
    BST(); //constructor
    bool empty();
    bool search(const T& item); //recursive search
    void insert(const T& item); //non-recursive insertion
    void remove(const T& item); //non-recursive
    int recursive_level(const T& item); //if the item does not
        exist on the tree, then return -1
    int nonrecursive_level(const T& item); //if the item does
        not
        exist on the tree, then return -1
    void leveltraversal(ostream& out);
    void recursive_preorder(ostream& out); //recursive
    void nonrecursive_preorder(ostream& out); // non-recursive
    void postorder(ostream& out); //non-recursive
    int height(); //recursive
private:
    class BinNode {
    public:
        T data;
        BinNode* left;
        BinNode* right;
        BinNode() : left(0), right(0) {}
        BinNode(T item): data(item), left(0), right(0) {}
    }; //end of BinNode
    typedef BinNode* BinNodePtr;
    BinNodePtr myRoot;
};
```

Your program must have the following:

1. A **constructor** to build an empty BST, an **empty()** to check if it is empty, recursive **search()** to search a given item, **insert()** an item (non-recursive), **remove()** an item (non-recursive), and an **output (operator<< is suggested)**. *All these functions can be found in textbook.* (6 points)
2. Add **BOTH** recursive and non-recursive **preorder** traversals as member function (*hint: you can use stack container provide by STL to do the non-recursive traversal*). (6 points)

```
void recursive_preorder(ostream& out);
void nonrecursive_preorder(ostream& out);
```

3. Write a recursive and non-recursive function member **level()** which determines the level of a given item in the tree. The root of the BST is at level 0, and its children are at level 1, and so on. (10 points)

```
int recursive_level(const T& item);
//if the item does not exist on the tree, then return -1
int nonrecursive_level(const T& item); //if the item does not
exist on the tree, then return -1
```

4. Write a member function **leveltraversal()** to traverse a tree level by level; that is, first visit the root, then all nodes on level 1 (children of the root), then all nodes on level 2, and so on. Nodes on the same level should be visited in order from left to right. (*Hint: Write a non-recursive function and use a queue of pointers, you can use STL queue container*). (5 points)

```
void leveltraversal(ostream& out);
```

5. Write a non-recursive member function **postorder()** to traverse a tree in post order. (Hint: using two stacks to keep track of nodes visited and the number of times a node is visited). (5 points)

```
void postorder(ostream& out);
```

6. Write a recursive member function **height()** to return the height of a tree. (5 points)

```
int height();
```