# Programming Assignments #2

## Due date: 10/19/2011 Wednesday 9:20AM

### 1. Problem (26 points)

In this assignment, you need to start from Lab4 (including bst.h, bst.cpp, test.cpp and Makefile) to complete t the tasks as described later.

In the lab4, you have already implemented

- Constructor
- Destructor
- bool empty()
- void insert(ElementType x)
- Three recursive traversals
- Nonrecurive preorder traversal
- Both recursive and non-recursive search operation

Before starting this assignment, you need to make sure all the above functions working correctly. If so, you will receive 5 points.

Then, you need to make BST to support the following operations. The member function prototypes are given below. You are NOT allowed to modify their prototypes!

- **void nonrecurs_inorder();**
  A non-recursive inorder traversal using STL stack. (3 points)
- **int level(const ElementType& item);**
  It determines the level of an given item on the tree. The root of the BST is at level 0, and its children are at level 1, and so on. Return -1 if the item is not on the tree. (3 points)
- **void level_traversal();**
  It traverses 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. *Write a non-recursive function. Use a queue of pointers, you can use STL queue*. (3 points)
- **int height();**
  It returns the height of the tree. (3 points)
- **ElementType getMin();**
  It returns the minimum data item on the tree. (3 points)
- **ElementType getMax();**
  It returns the maximum data item on the tree. (3 points)
- **void remove(const ElementType& x);**
  It deletes the specified element from the tree if it exists. (3 points)

## 2. References

[1] STL stack: http://www.cplusplus.com/reference/stl/stack/
[2] STL queue: http://www.cplusplus.com/reference/stl/queue/

For example, if you want to use a STL queue to store integers, you can declare a STL queue like this:

```
// queue::push/pop
#include <iostream>
#include <queue>
using namespace std;

int main ()
{
	queue<int> myqueue;
	int myint;

	cout << "Please enter some integers (enter 0 to end):\n";

	do {
		cin >> myint;
		myqueue.push (myint);
	} while (myint);

	cout << "myqueue contains: ";
	while (!myqueue.empty())
	{
		cout << " " << myqueue.front();
		myqueue.pop();
	}

	return 0;
}
```

More examples about these two containers can be found in the links.

## 3. Submission

Before submission, you should ensure your program has been compiled and tested extensively. Your assignment receives zero if your code cannot be compiled and executed. Collaboration is prohibited!

You are encouraged to include a readme file in your submission, which should state your program's purposes, assumptions, and issues (if any).

You can submit your program multiple times before the deadline. The last submission will be used for grading.

To submit your assignment, you should follow the steps below:

a. Wrap all your files into a package, named ***hw2.tar***
   **tar -cvf hw2.tar bst.h bst.cpp test.cpp Makefile readme**
b. Submit your newly generated package ***hw1.tar***
   **/home/fac/testzhuy/CPSC250/SubmitHW hw2 hw2.tar**

## 4. Grading

- You will receive zero if your program cannot be compiled and executed or if your submission does not include the required files.
- The lab exercises receive 5 points in total. If you have any mistakes on the lab exercises, you receive 0- 80% of the points.
- Each function except the lab exercises receives 3 points. If you did not test it in your test.cpp, you receive 0-50% of the points. If your function has problems, you receive 0-%80 of the points

## 5. Testing

As mentioned earlier, you need to test your program extensively before submission. In order to make your program testing to meet the minimum standard, I provide a test case below for your testing purpose. Note that this does not cover all use cases.

```
ElementType  A[] = {23,  45, 12, 35, 60, 99, 6, 87, 9, 10, 14, 25 };
BST    bt1;
//test insert()
for (int i = 0; i < 12; i++)
     bt1.insert(A[i]);

cout << "\nNon-recursive inorder travels: \n";
bt1.nonrecurs_inorder();  //test nonrecursive inorder

cout << "\nLevel  traversal: \n";
bt1.level_traversal();

cout << endl;
cout << "The min data item is: " << bt1.getMin() << endl;
cout << "The max data item is: " << bt1.getMax() << endl;
cout << "The tree height is: " << bt1.height() << endl;

//test deletion
bt1.remove(45);
bt1.remove(8);
cout << "\nNon-recursive inorder travels: \n";
bt1.nonrecurs_inorder();  //test nonrecursive inorder

cout << "99 is on the level of " << bt1.level(99) << endl;
cout << "45 is on the level of " << bt1.level(45) << endl;
```