Data Structures C & D
FAST-NU, Lahore, Fall 2016

Homework 5

Binary Search Trees

Due: Thursday Oct 5 11:55PM Marked out of 50 points.

All the code should be written in a single .cpp file.

1. Complete the implementation of the class template **bst** given below:

```
template <class T>
class bst{
private:
    struct Node{
        T data;
        Node * lchild, * rchild;
    };
    Node * root;
    int n;
private:
    Node * createNode(const T& obj);
    void getNodes_inOrder(vector<T>&nodes);
    void getNodes_preOrder(vector<T>&nodes);
    void getNodes_postOrder(vector<T>&nodes);
    int height(Node*);
public:
    bst();
    void insert(const T & obj);
    void remove(const T & obj);
    bool search(const T & obj);
    int size();
    int height();
};
```

corrections: (1) the getNodes_inOrder and other traversal functions should also accept a Node * n pointer. (2) The type of the vector passed by reference to these functions should be vector<Node*> not vector<T>

2. **A) Add an iterator to this class**, just like we did for the class LinkedList. In this case, after an iterator has been added we can write a loop like the following to traverse through the data in the tree in **sorted** order. In the following, t is a bst of type int.

```
for(bst<int>::iterator itr=t.begin(); itr!=t.end(); itr++){
    // *itr returns the value in particular node
}
```

To enable this you will add a class iterator to the bst class, with the following data-members.

```
public://this is inside class bst
    class iterator{
        vector<Node*> & nodes;
        int index;
    public:
        iterator(const iterator& obj);
        bool operator != (iterator& obj);
        iterator & operator ++ ();
        iterator operator ++ (int);
        T operator * ();//dereference
    };
```

Methods begin and end are added to the class bst as well. The method begin will use the private method getNodes_inOrder to fill the vector inside iterator and on each call to operator ++ of iterator the index inside iterator will be incremented. The method end will empty the vector only then index reaches the end of the vector. Figure out the rest for yourself.

3. Now we shall add a few interesting **methods to the class BST**. The user of your program should be able to use these methods from a “menu” provided on the console. The menu should also provide options to print the tree in sorted order.

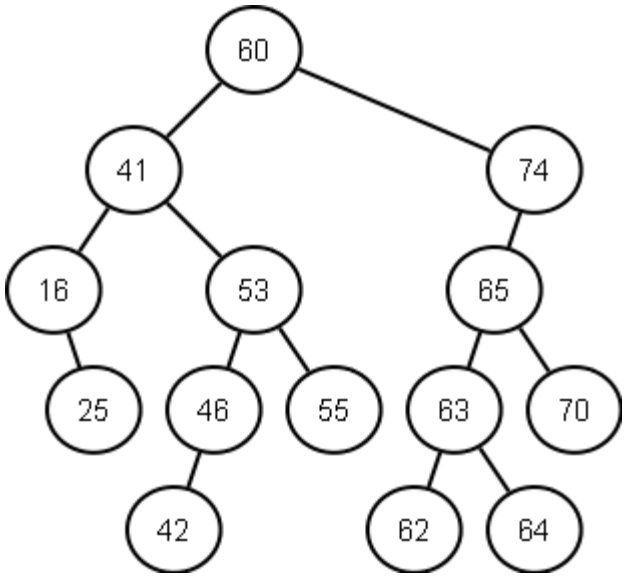

Figure I

i) **int width():** returns the number of nodes in the most full level of the tree (the highest number of nodes in any level). For example, the width of the tree in Figure I is 5.

ii) **void pathSum(vector<T> & sums):** puts the sums of all nodes in each path of the tree in the vector sums. The total number of sums is equal to the number of unique paths in the tree, in other words, the number of leaves in the tree. The sums for the tree in Figure I are (left to right): 142, 242, 204, 324, 326 and 269. The menu option print path sums should print these sums.

iii) **void updateKey(const T & key1, const T& key2):** finds the node with key1 and changes it to key2. This may disturb the bst property (of having smaller things to the left and bigger to the right); the function makes sure the problem is removed by appropriately making changes (as few changes as possible).

4. Now we'll perform a small experiment to see how well a bst performs if data is added to it at random. As you know, for a bst on n nodes the best possible height is lg(n+1) (log of n base 2), and the worst is n-1. Write code to do the following:

i. Generate 1000 random bsts (one at a time) by generating 1000 random numbers each time and inserting them into a bst. (so n = 1000 for each of the 1000 trees).
ii. Compute the height of each tree by using the height function. Take the average height, h/1000.

iii. Compare this value with lg(1000+1) and 1000-1, where it does lie in this range? What does the result suggest about bsts created with random data?

iv. Based on your findings, if you were asked to write a function to read values into a bst from a file on the disk, how will you read these values?

THE END