Assignment 5

CS-AD-103: Data Structures

Assignments are to be submitted in groups of two or three. Upload the solutions on NYU classes with one PDF file for the theoretical assignments and separate C++ files for each coding assignment.

Problem 1 (20 points).
Implement a Binary Search Tree class which supports the following operations: find, insert, remove, depth and printPreorder. The class declarations are given below - *please don't change it.* You need to define the functions in the BST class appropriately. You may add other private functions or variables to the BST class if you need to.

```
template <typename S>
class Node{
 public:
        S key;
        Node<S> *left, *right, *parent;
};

template <typename T>
class BST{
private:
      Node<T> *root;

public:
    BST();
    ~BST();
    void insert(T key);
    void remove(T key);
    Node<T>* find(T key);
    void printPreorder();
    int depth();
};
```

Below follows a description of what the public functions (other than the constructor and destructor) in the BST class are supposed to do:

- `insert` takes a `key` and inserts it into the binary tree.
- `remove` takes a `key` and deletes it from the tree (if it exists).
- `find` takes a `key` and returns a pointer to the node with that key value if there is such a node, otherwise it returns `NULL` (or `nullptr`).
- `printPreorder` prints the keys in the tree according to the preorder traversal.
- `depth`() returns the depth of the current tree.

Please use the algorithms discussed in the class for your implementations and check your code by trying various operations in different orders.

Problem 2 (10 points)**.**
A **full binary tree** is a binary tree in which each node has either two children or no children. There is a full binary tree that has the following traversal orders:

Preorder traversal: F G A K E H L J I M D C B
Postorder traversal: A K G L I D C M J H B E F

Reconstruct the tree and explain in as much detail as possible how you did it. Please verify your answer.

Problem 3 (10 points)**.** Suppose that you are given the pointer to the root of a binary tree, each of whose nodes store a `key` which is an integer. Give a linear time algorithm, in terms of the number of nodes in the tree, to check whether the tree is in fact a binary search tree with respect to the keys.