```
/* SplayTree.h
 * Tahmid Rahman
 * Dylan Jeffers
 *
 * This is the header file
 * for SplayTree
 */

#ifndef SPLAYTREE_H_
#define SPLAYTREE_H_

#include "BST.h"
#include "pair.h"
#include "library/queue.h"
#include <iostream>

using namespace std;

// Forward declaration of SplayTreeNode class
template <typename K, typename V> class SplayTreeNode;

template<typename K, typename V>
class SplayTree: public BST<K,V> {
    private:
    int size;
    SplayTreeNode<K,V>* root;

    public:
        SplayTree();
        ~SplayTree();

        /* All public functions declared/detailed in BST.h */
        /* The following methods are defined in SplayTree-inl.h */
        int getSize();
        bool isEmpty();
        int getHeight();
        K getMax();
        K getMin();

        /* dictionary operations */
        void insert (K key, V value);
        void update (K key, V value);
        bool contains (K key);
        void remove (K key);
        V find (K key);

        /* traversal operations */
        Queue< Pair<K,V> >* getPreOrder();
        Queue< Pair<K,V> >* getInOrder();
        Queue< Pair<K,V> >* getPostOrder();
        Queue< Pair<K,V> >* getLevelOrder();
        K getRootKey();

    private:
            /*declarations of our interal private methods */
        SplayTreeNode<K,V>* insertInSubtree(SplayTreeNode<K,V>* current, K key, V
value, bool* inserted, bool* skip);

        void updateInSubtree(SplayTreeNode<K,V>* current, K key, V value);
        SplayTreeNode<K,V>* removeFromSubtree (SplayTreeNode<K,V>* current, K key);

        bool containsInSubtree (SplayTreeNode<K,V>* current, K key, bool * skip);

        K  getMaxInSubtree(SplayTreeNode<K,V>* current);

        K getMinInSubtree(SplayTreeNode<K,V>* current);

        void buildPreOrder (SplayTreeNode<K,V>* current, Queue< Pair<K,V> >* it);

        void buildInOrder  (SplayTreeNode<K,V>* current, Queue< Pair<K,V> >* it);

        void buildPostOrder(SplayTreeNode<K,V>* current, Queue< Pair<K,V> >* it);

        void traverseAndDelete (SplayTreeNode<K,V>* current);

        SplayTreeNode<K,V>* splay(SplayTreeNode<K,V>* current, K key);

        int getHeightOfSubtree(SplayTreeNode<K,V>* current);

        /* the six rotations needed to fix each of the six imbalances*/

        SplayTreeNode<K,V>* rightRotate(SplayTreeNode<K,V>* current);
        SplayTreeNode<K,V>* leftRotate(SplayTreeNode<K,V>* current);

        SplayTreeNode<K,V>* rightLeftRotate(SplayTreeNode<K,V>* current);
        SplayTreeNode<K,V>* leftRightRotate(SplayTreeNode<K,V>* current);

        SplayTreeNode<K,V>* rightRightRotate(SplayTreeNode<K,V>* current);
        SplayTreeNode<K,V>* leftLeftRotate(SplayTreeNode<K,V>* current);
};



/* SplayTreeNode is templated class that stores data for each node in the
SplayTree */

template <typename K, typename V>
class SplayTreeNode {
    private:
        K key;
        V value;

        /*using parent implementation vs record implementation (ask me if
         if you're unsure what this means */
        SplayTreeNode<K,V> * left;
        SplayTreeNode<K,V> * right;

        SplayTreeNode();
        SplayTreeNode(K k, V v);
        int getHeight();
    //so SplayTree can directly access private aspects of this class
    friend class SplayTree<K,V>;
};


#include "SplayTree-inl.h"
#include "SplayTree-private-inl.h"
```

```
#endif
```