```
/*SPLAVL.h
 *Tahmid Rahman
 *Dylan Jeffers
 *
 *SPLAVL header file
 */

#ifndef SPLAVL_H_
#define SPLAVL_H_

#include "BST.h"
#include "pair.h"
#include "library/queue.h"
#include "math.h"

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

template<typename K, typename V>
class SPLAVL: public BST<K,V> {
    private:
    int size;
    SPLAVLNode<K,V>* root;
    int currentCount, maxCount;
    int currentRatio, maxRatio;


    public:
        SPLAVL();
        SPLAVL(int maxC, int maxR);
        ~SPLAVL();

        /* All public functions declared/detailed in BST.h */
        /* The following methods are defined in SPLAVL-inl.h */
        int getSize();
        bool isEmpty();
        int getHeight();
        K getMax();
        K getMin();
        void setMaxCount(int maxC);
        void setMaxRatio(int maxR);
        bool isBalanced();

        /* 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:
            /*declare our interal private methods */
```

```
        bool isBalancedInSubtree(SPLAVLNode<K,V>* current);
        SPLAVLNode<K,V>* insertInSubtree(SPLAVLNode<K,V>* current, K key, V value);

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

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

        //SPLAVLNode<K,V>* findInSubtree(SPLAVLNode<K,V>* current, K toSplay);

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

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

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

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

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

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

        void computeHeightFromChildren(SPLAVLNode<K,V>* current);

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

        /* the four rotations needed to for SPLAVL type splays*/

        SPLAVLNode<K,V>* rightRotate(SPLAVLNode<K,V>* current);
        SPLAVLNode<K,V>* leftRightRotate(SPLAVLNode<K,V>* current);
        SPLAVLNode<K,V>* leftRotate(SPLAVLNode<K,V>* current);
        SPLAVLNode<K,V>* rightLeftRotate(SPLAVLNode<K,V>* current);
};



/* SPLAVLNode is templated class that stores data for each node in the
SPLAVL */

template <typename K, typename V>
class SPLAVLNode {
    private:
        K key;
        V value;
        //adding height here since we will definitely need it for SPLAVL portion
        int height;
        /*using parent implementation vs record implementation (ask me if
         if you're unsure what this means */
        SPLAVLNode<K,V> * left;
        SPLAVLNode<K,V> * right;

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

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

#endif
```