Homework # 03 *Due: 19 Nov 2024*

Exercises

Write a generic data type for a Right Leaning Red Black Tree and a word dictionary RLRB implementation. The goal of this assignment is to solidify understanding of RLRBs and their practical applications.

Exercise 1.....

Create a generic data type *RLRB* that implements the following API:

```
template <typename Key, typename Value>
class RLRB {
public:

    RLRB(); // Constructor

    ~RLRB(); // Destructor

    void put(const Key& key, const Value& value); // Insertion

    Value get(const Key& key); // Retrieval

    void remove(const Key& key); // Deletion

    void inorderTraversal() const; // Inorder traversal

    void preorderTraversal() const; // Preorder traversal

    void postorderTraversal() const; // Postorder traversal

    void printRange(const Key& lo, const Key& hi) const; // Prints values in the given range

    Value findMin() const; // Finds the minimum value

    Value findMax() const; // Finds the maximum value

    int height() const; // Calculates the height

    bool isEmpty() const; // Checks if the Tree is empty

    void clear(); // Clears the Tree

private:

    Node* root; // Root node of the tree
```

```
    // Node class
    class Node {
    public:
        static const bool RED = true;
        static const bool BLACK = false;
        Key key;
        Value value;
        bool color; // color of parent link
        Node* left, right;
        Node(const Key& key, const Value& val);
        bool isRed(Node* h);
    };

    // Private Helper Functions
    void rotateLeft(Node* node); // Performs one left rotation on a node
    void rotateRight(Node* node); // Performs one right rotation on a node
    void flipColors(Node* node); // Flips colors of parent and children links
    Node* put(Node* node, Key& key, Value& val);  // Recursive insertion
    void inorderTraversal(Node* node) const; // Recursive inorder traversal
    void preorderTraversal(Node* node) const; // Recursive preorder traversal
    void postorderTraversal(Node* node) const; // Recursive postorder traversal
    void printRange(Node* node, const Key& lo, const Key& hi) const; // Range retrieval
    Value findMin(Node* node) const; // Finds min in subtree
    Value findMax(Node* node) const; // Finds max in subtree
    int height(Node* node) const; // Calculates height of subtree
};
```

Corner cases: Throw a std::runtime_error if the client calls either of the retrieval functions (like search, delete etc.) when the RLRB is empty.

Unit testing: Your RLRB::unit_test() method must call directly every public constructor and method to help verify that they work as prescribed by printing results to standard output.

Exercise 2……..

Write a program to store a word dictionary in a *DictionaryRLRB*, where each *DictionaryNode* contains a *word* and its *meaning*:

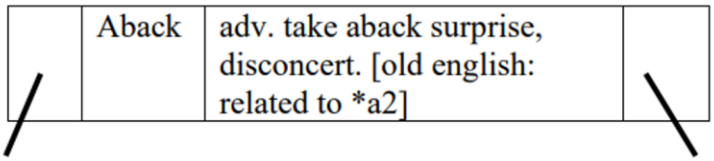
Use rules of RLRB as:

a. Every word in n's left subtree is less than the word in node n.
b. Every word in n's right subtree is greater than or equal to the word in node n.

Implement the basic methods of Right Leaning Red Black Tree that includes: *put*, *get*, and *traverse*.

The definition of a tree node should be defined as shown:

```
class DictionaryNode {
public:
   std::string word;
   std::string meaning;
   Node* left, right;
   …
};
```



The dictionary is available as a comma-separated values (csv) file with this assignment. Read the **dictionary.csv** file line by line and split each line into two parts: word and its meaning.

Unit testing: Your DictionaryRLRB::unit_test() method must call directly every public constructor and method to help verify that they work as prescribed by printing results.

……

Submission

Submit the programs **RLRB.cpp** and **DictionaryRLRB.cpp**. Finally, answer the questions provided in **readme_a3.txt** file and submit the file.

Grading

file	marks
RLRB.cpp	10
DictionaryRLRB.cpp	10
readme_a3.txt	5