Maximum points: 200. Individual Work Only.

Due Date: Oct 23, 2024, before 11:59pm (late submissions will get a score of zero)

Objectives

- Implement the *SymbolTable* class in C++ using Red-Black Trees
- Use the *SymbolTable* class to implement the word frequency problem

Problem Description

1. Implement the *SymbolTable* class using the given header file *ST.hpp* and test your implementation using the driver program *STMain.cpp*. Use the Red-Black tree implementation provided to implement the *SymbolTable* class. Note that you should not modify any code from the Red-Black tree implementation, the only file you must update is *ST.hpp*. Also, make sure that the implementation of each method conforms with the specified algorithmic complexity. The table below provides the method name, description, and expected worst-case complexity for each method:

Name	Description	Time Complexity	Points
ST()	Default constructor	Constant	5
~ST()	Destructor	Linear in size of ST	5
[] operator	Inserts a key if it is not already present and returns a reference to the value corresponding to that key	Logarithmic in size of ST	20
insert(key, value)	Inserts the given (key,value) pair if the key does not exist, otherwise will replace current key with new value	Logarithmic in size of ST	10
remove(position)	Remove element at the given position	Amortized constant	10
remove(key)	Remove element with given key	Logarithmic in size of ST	10
clear()	Remove all elements from the ST, after this size() returns zero	Linear in size of ST	10
empty()	Checks if ST has no elements	Constant	5
size()	Returns no. of elements in ST	Constant	5
count(key)	returns no. of elements that match key (0 or 1 is returned as keys are unique)	Logarithmic in size of ST	10

find(key)	finds a node with matching key and returns iterator to the matching element (nullptr when no match found)	Logarithmic in size of ST	10
contains(key)	Returns true is the key is found in ST, otherwise false	Logarithmic in size of ST	10
toVector()	Returns contents of ST as a vector of (key,value) pairs	Linear in size of ST	10
displayTree()	Prints the Red-Black Tree associated with ST	Linear in size of ST	0
display()	Prints ST as key: value	Linear in size of ST	10

2. Use the SymbolTable class to implement the solution to Homework-1 and compare the performance of the program with that of Homework-1 and complete the table below:

Input File	# words	Time Taken Homework-1	Time Taken using SymbolTable
Input1.txt			
Input2.txt			
Input3.txt			

3. Include the table in the report and write a short description that compares the performance of the two versions and explain any differences between the two approaches as well as any differences in the execution time.

Program Documentation and Testing

1. Use appropriate variables names and indentation in your source code.
2. Include meaningful comments to indicate various operations performed by the program.
3. Programs must include the following header information within comments:

```
/*
   Name:
   Email:
   Course Section: Fall 2024 CS 201
   Homework #:
*/
```

Submission

Upload only the source files (.h or .hpp or .cpp or .cc files) and the report (Word or PDF file) to Blackboard in the assignment submission section for this homework. There is no need to upload the files related to Red-Black tree. Do not upload zip/tar files to Blackboard, upload individual source files (no object files or executables) and the Word or PDF file for the report.

Grading Rubrics

The program will be graded not only for correctness with the sample input provided but also for implementing each method with the specified time complexity. The following rubrics is used for grading:

Description	Points
1. Correct implementation of ST (see table above for detailed breakdown)	130
2. Implementation of word count frequency using ST	50
3. Report with table and performance comparison	20