CS515

Assignment 11

The purpose of this assignment is to give you practice with balanced search trees. You will implement the Map ADT with an underlying AVL search tree.

Download all files from `~cs515/public/11P`

In Assignment 9, you wrote the Map ADT similarly to the std::map, but the underlying tree is not balanced. To guarantee O(logN) performance, you must upgrade the BST into an AVL tree.

You need to create a header file, map.h, based on Assignment 9 and according to your current implementation. The **public interface** of the Map container must remain the same as in Assignment 9, with only one exception: the erase() method is not required. Meanwhile, other Map operations, such as the constructors and destructor, the one-directional Map iterator, operator[], must all be supported. The Map container must be generic and implemented as a class template, so it can be instantiated as `Map<std::string, int>`, for example. You may add other methods as you see fit.

Make sure you keep the overloaded operator << and printTree() methods provided in the map.cpp file unchanged, in order for the graders to verify your results. A good starting point for this assignment is to take what you did for Assignment 9P, add pieces from Lab 9 where you had a simple AVL tree written under the hood of a Set ADT, and make some modifications for threads in the rotation/height code.

To test your program, you can reuse the test cases for Assignment 9, excluding the tests for erase() method. You should check the tree printout and verify the AVL tree is correct.

Note that the std::map() container can only be used as in the starter code of Assignment 9. Other usages of STL containers are not allowed. In addition, you must implement an AVL tree in this assignment, not any other types of balanced search tree (for example, a Red-Black tree).

Submission:

- Submit **`map.cpp`** and **`map.h`.**
- You should also fill out the README file and submit it as well. Submit the following files to the appropriate assignment on Mimir:

  `map.cpp map.h README`
- Do not turn in executables or object code. Programs that produce compile time errors or warnings will receive a zero mark (even if it might work perfectly on your home computer).
- Be sure to provide comments in your program. You must include the information as the section of comments below:

```
/**     CS515 Assignment 11
File: XXX.cpp
Name: XXX
Section: X
Date: XXX
Collaboration Declaration: assistance received from TA, PAC etc.
*/
```

Some notes on grading:

- Programs are graded for correctness (output results and code details), following directions and using specified features, documentation and style.
- To successfully pass the provided sample tests is not an indication of a potential good grade; to fail one or more of these tests is an indication of a potential bad grade.
- You must test thoroughly your program with your own test data/cases to ensure all the requirements are fulfilled. We will use additional test data/cases other than the sample tests to grade your program.
- Here is a tentative grading scheme.

map test run 1	15
map test run 2	15
map test run 3	12
map test run 4	12
map test run 5	12
map test run 6	12
map test run 7	10
map test run 8	12

each test includes 3 pts for valgrind check except for test run 7