CS515

Assignment 9

The purpose of this assignment is to give you practice with implementing a Map ADT similar to the basic functionalities of std::map in the C++ standard template library.

Download all files from `~cs515/public/9P`

You are asked to implement a generic Map ADT using a right-threaded binary search tree. The Map should be implemented as a class template. The header file is given in the starter code. You will need to finish the implementation in `map.cpp`. You can reuse the provided methods without modification (unless you found any bugs in them.)

Since you are writing a template class, the compilation line below will ***not*** work.

```
g++ -c map.cpp
```

To test your template implementation, you need to write a test driver that includes the template header `map.h`. Then, compile the driver file directly. For example, given the simple test driver maptest.cpp, you should compile it directly

```
g++ maptest.cpp
```

The included class templates are instantiated during the compilation and then compiled with the driver. **The class templates cannot be compiled individually.**

The `map.h` file provides declarations for all the methods that need to be implemented. In particular, the following are required:

- A **Map iterator** that supports one-directional (++) iteration. The Map is implemented as a sorted container so the iterator traverses all elements in order. Note that the container is a right-threaded tree that uses a dummy root sentinel.
- An **`erase()`** method to remove an element from the Map. The method removes an existing node while maintaining the structure of the underlying threaded tree. This operation could be tricky to code, as you don't want to "break" the thread while removing a node. One approach to do this is to consider the following four cases of node removal: removing a node with no child links, with left child link only, with right child link only and two children links.
  You must implement the map erase() method following the "delete by swapping algorithm" used in Assignment 7. (The node to be deleted is swapped with its successor.)
- An assignment operator that produces a deep copy of the threaded tree.

You should not modify the header `map.h` file, and the `map.cpp` file should only include the `map.h` header file during early testing (before you turn it into a template).

The test cases used for Assignment 7 can be reused with minor modification and expanded. You just need to create a template instantiation rather than using an actual class. You should be sure to add test cases for testing iterator traversals. Make sure to test thoroughly with your own additional test cases before submission. Note that we will not be grading your tests directly this time.

Note that although the starter code is given in a templated version, it is highly recommended to start your implementation with a non-templated threaded BST tree, and later transform the code into a template. This will help you avoid template-class-related compile time errors and focus on the logic of the tree operations first. You may want to keep a non-templated version of the assignment around as well, in case a later assignment builds off of this code!

Submission:

- Submit **`map.cpp.`**
- 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 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 9
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.

	Test itself	Valgrind
map test run 1	8	2
map test run 2	8	2
map test run 3	10	2
map test run 4	10	2
map test run 5	10	3
map test run 6	10	4
map test run 7	10	4
map test run 8	10	5
Others		
Use of STL containers (beyond starter code's use)	-100	
Fail to implement the given delete algorithm	-40	