Computer Science II — Homework 8 — cs2multiset

This assignment is due **Friday, April 13th** at 11:59:59pm and is worth 80 points toward your homework grade.

Your job is to create a version of the `std::multiset<T>` container class from the standard library that we will call `cs2multiset<T>`. This container differs from the `std::set<T>` container in that multiple instances of an entry are allowed in the set. It is important to realize that this does not mean that you can simply have a counter with the number of occurrence of a particular entry. This is because equality is defined by `operator==` and two objects can be equal according to this operator, but may still different in content.

You must use a binary search tree implementation, and you must start from the template file, `cs2multiset.h` provided on the course web site. You may not remove code from this template, but you must, of course, add to it. We have provided a `test_multiset.cpp` function on the course website, and you should use this as the basis for testing your implementation. This is the code that the submission script will run, but the grading scripts may be more extensive.

The functions you must implement are explained in `cs2multiset.h` as well as in the example test main program and its output. In `cs2multiset.h` they are highlighted by

```
// ****
```

You may add any number of functions, primarily (even exclusively) private member functions called by the public member functions already declared.

Here are some important points to note:

- The required functions include the increment and decrement operators on iterators; you should base your implementation of these iterators on the notes from Lecture 18 on finding the in-order successor of a tree node. In particular, **pay careful attention to maintaining parent pointers** during insert, copy and erase functions.
- One problem we will have in testing and grading this assignment is that a fixed sequence of insert and erase operations can result in trees with different structures. We would like to try to minimize these differences. Therefore, please use the following rules:
  - When searching for a location to insert a value `x`, move down the right subtree whenever `x` is equal to the value stored at the node.
  - During erase, when a node has two children, use the smallest value (the leftmost descendent) in the right subtree of the node as the replacement value for the node.
- As you will see, you are asked to implement two `erase` functions, one erasing the value at a particular iterator and the other erasing all of the entries in the multiset having the particular value. These are not easy to get correct. They are also not easy to make efficient. Do your best, and explain your technique in the `readme.txt` file.

Your submission zip file should only include the `readme.txt` file and the `cs2multiset.h` file.