EECS 560 Lab 8: Leftist Heap

Due date
10/25/2020 Sunday, 11:59 pm

Objective
Get familiar with leftist heap ADT implementation with C++. Work on basic operations of leftist heap.

Specification of the ADT
Refer to section 6.6 in Textbook. Implement the code for methods in figure 6.25 of leftist heap class skeleton, which has Type declaration of functions of leftist heap. The implementation for some of these methods available as mentioned below:

1. Figure 6.26, Figure 6.27: Public member functions calling private recursive member function of leftist heap to implement the **Merge** routine.
2. Figure 6.29: Public member functions to **Insert** one element into leftist heap.
3. Figure 6.30: Public member functions for **deleteMin**.

Requirements to completed from text from template

1. Change the **Minimal** leftist heap class into "**myLeftistHeap**".
2. Implement **myLeftistHeap( )** defined in figure 6.25, the constructor with default behavior.
3. Implement **myLeftistHeap(const myLeftistHeap & rhs)** defined in figure 6.25, the copy constructor.
4. Implement **myLeftistHeap(myLeftistHeap && rhs)** defined in figure 6.25, the copy constructor.
5. Implement **~myLeftistHeap( )** defined in figure 6.25, the destructor that releases all allocated memory.
6. Implement public member functions **myLeftistHeap & operator=(const myLeftistHeap & rhs)** defined in figure 6.25, enabling the assign operator for your leftist heap.
7. Implement public member functions **myLeftistHeap & operator=(myLeftistHeap && rhs)** defined in figure 6.25, enabling the assign operator for your leftist heap.
8. Implement public member functions **bool isEmpty( ) const** defined in figure 6.25, which tells if your leftist heap is empty or not.
9. Implement public member functions **const Comparable & findMin( ) const** defined in figure 6.25, that returns the minimal element in your leftist heap.
10. Implement public member functions **void makeEmpty( )** defined in figure 6.25, that deletes all elements from your leftist heap.
11. Implement public member functions **void printInOrder( )** that prints the heap using **in-order traversal**.
12. Implement public member functions **void printLevelOrder ( )** that prints the heap using **level-order traversal**.
13. Any other internal (private) functions you need to implement the previous functions.

Testing and Grading

We will test your implementation using a tester main function, on a number of instances that are randomly generated. We will release the tester main function, several instances (will be different from the grading instances but will have the same format), and expected output together with the lab instruction via Blackboard. You code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

The command line we are going to use for compiling your code is: "g++ -std=c++11 main.cpp" (note that main.cpp will try to include the .hpp file you submit, and your .hpp file needs to be property implemented to compile successfully).

Your final score will be the percentage your program passes the grading instances. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline

Please submit your implementation as one .hpp file, with file name "myLeftistHeap_[YourKUID].hpp". For example, if my KU ID is c123z456, my submission will be a single file named "myLeftistHeap_c124z456.hpp". Submissions that do not comply with the naming specification will not be graded. Please submit through Blackboard.