ECE 2574 ♦ Introduction to Data Structures and Algorithms
Homework 5: Merge Sort

Overview: For this assignment, you must write a C++ program, which sorts a set of data items in the ascending order with respect to a data field using the *merge sort* algorithm.

Objectives: The main goals of this assignment are to learn about the basic sorting algorithms and data structures such as binary trees, tree traversal algorithms, and their implementations.

Program specifications: Write a C++ program. The program implements a pointer-based Merge Sort algorithm. This algorithm can be found in the textbook.

Please name your program as **`mergesort`**. Your program should be invoked as follows:

`mergesort` *`input_file output_file`*

The alphabets and their frequencies should be taken from **`input_file`** specified on the command line, which contains the alphabets and their frequencies, one per line, in random order. The resulting **`output_file`** must contain them in the ascending order of their frequencies. For example, consider the following files:

testfile1_in.txt	**testfile1_out.txt**
a 45	f 5
f 5	e 9
c 12	c 12
e 9	b 13
d 16	d 16
b 13	a 45

The file **testfile1_in.txt** contains the characters and their frequencies, **testfile1_out.txt** contains the sorted file.

File Naming Conventions: Use the following file names**:** `mergesort.h`, `mergesort.cpp` (implementation file), and `sortlist.cpp` (client file). You may use additional files as well, depending on how you decide to subdivide the code. Be sure to follow good programming practices, as described in the *Programming Style Guidelines* handout.

Assumptions: You are required to implement a pointer-based binary tree. You may write such an ADT from "scratch," or implement the ADT Binary Tree described in the textbook (see Chapter 15). *Note that the use of STL is not allowed.*

Exception Handling: Your implementation should handle the following exceptions:

1. If input and/or output filenames do not appear on the command line, or the input file does not exist.
2. Bad input file. This includes an input file that contains a line with a missing character or a line with a missing frequency, and other incorrectly formatted files (within reason).

3. If new returns NULL when allocating memory.

What to hand in: Upload your source files (`mergesort.h`, `mergesort.cpp`, and `sortlist.cpp`) to Scholar. If you have more files than Scholar allows, combine them into a single ZIP file. Be careful to use these file names, to test your files *before* uploading them, and to upload the *correct* files. The files that you upload to Scholar are the files that will be graded.

Partial credit: If you could not get your program to run, also turn in a written description (MS Word or PDF file) that describes 1) the problems you encountered, 2) what parts of your program work correctly, and 3) suggestions to the grader as to how your program may be tested for partial credit consideration. If your program runs, you do not need to turn in such a file.

Grading: The assignment will be graded out of a maximum of 100 points: 90 points depend on correct program operation, and the remaining 10 points are based on the programming style.

- The grader will compile and link your code, and will run it with different test case files. We will try to automate part of this process. If your code does not compile, then the grader will be tempted to assign 0 points for correct operation, and then move on to the next student's submission. It is therefore very important that you upload the final, correct versions of your files with the correct file names.
- Programming style includes clarity, comments, indentation, error handling, etc., as described in the *Programming Style Guidelines* handout.

Honor Code: You must work independently on this assignment. Please review the statement in the syllabus.