Data Structures and Algorithms I
Spring 2018
Programming Assignment #2

You are going to complete a program that sorts the nodes of a linked list. The program will load data from an input file specified by the user and create a linked list of pointers to data objects using the provided C++ list class. Each data object will consist of four fields. The first two fields will be unsigned 32-bit integers (but these two fields will be generated in different manners, as described later). The third field will be a single printable, standard ASCII character. The fourth field will be a C++ string consisting of exactly 25 printable, standard ASCII characters.

After creating the list, the program will sort the list according to one of the four fields, as specified by the user. The sorted list will then be written to an output file. The input and output files will have the same format. The first row will be an integer indicating how many rows follow. Each row after that represents a single data object, with the values of four fields separated by single spaces. There will be no leading or trailing whitespace, and each row will be followed by a Unix-style newline character ('\n'). If items in the column being sorted are identical, their relative positions to each other should not change; i.e., the sort must be stable.

I am providing you with code that handles most aspects of the program, and *you may not make any changes to the provided code or its behavior* (this will be explained further in class). The provided code includes the implementation of a simple class to store the data objects, the file loading routine that loads the data from an input file, and the file saving routine that writes the sorted data to an output file. There is also a call to a sort routine that you must fill in. You may also add additional functions, additional class definitions, or additional global variables if you wish, but all the added code must be included below a specific comment which indicates that the code above the comment may not change. (You should even be able to include additional provided header files below the comment if you wish. If you want to do this but your compiler does not support it, then include them at the top of the file, and mention this in your e-mail when you submit the program.) I may use the "diff" command to make sure you have not changed any code above the comment.

I will run your program four times, to sort the data from a single test file based on each of its four columns. The test file will contain approximately (within 1 percent of) 1,000,000 data objects. The fields of the data objects will be generated as follows:

- The first field of the i^{th} data object will be i + 10 + [random offset from -10 to 10]. Therefore, if you look at the first column of the input file (not including the first row, which specifies the number of rows that follow), it will be approximately sorted.
- The second field of the i^{th} data object will be a randomly generated unsigned 32-bit integer. Each bit of the field will be generated independently.
- The third field of the i^{th} data object will be a randomly generated, printable, standard ASCII character, not including space (i.e., an ASCII character in the range of 33 to 126).
- The fourth field of the i^{th} data object will be a randomly generated string containing exactly 25 printable, standard ASCII characters, not including space (i.e., ASCII characters in the range of 33 to 126); each character of the string will be generated independently.

Every working program will be assigned a score that is based on the CPU times that the program takes to sort the test data based on each of the four fields. If $time_1 \ldots time_4$ are the CPU times required by the program when sorting based on each of the four fields, the overall score for the program will be $time_1 + time_2 + time_3 + time_4$ (i.e., the sum of the four times). Assuming that the program works (i.e., it generates the correct output for all four sorts), *your program will be graded almost entirely based on this overall score just described*. I expect working programs; penalties for non-working programs will be discussed in class. I reserve the right to take off a few points for poor indentation (which does not affect speed, since it is ignored by the compiler), but otherwise, you will not lose points for lack of elegance. Almost anything goes, as long as your write the program individually and do not violate any of the rules described here or in class. You may use any provided classes or routines to which you have access, including the provided sort member function of the C++ list class (which provides an implementation of merge sort) or the provided sort function from the C++ algorithm library (which provides an implementation of a modified quicksort). However, *you may not write code that changes the behavior of the provided code* – this will be explained further in class.

I will not provide my test data in advance. However, I will provide sample data that was generated the same way as my test data, and you can use this for testing. Furthermore, I will allow pre-submissions, and I will let you know roughly how your program is performing on my actual test data. To be evaluated, pre-submissions must arrive at least 24 hours before the program deadline. Expect that it may take me up to 24 hours (or sometimes longer) to reply to a pre-submission. You may send multiple pre-submissions, but only one at a time (i.e., if you send a pre-submission before I replied to your previous pre-submission, I will only evaluate one.)

Note that you are allowed to submit programs that are not theoretically guaranteed to sort the data correctly! If your program outputs the wrong thing, but you can convince me that the chance of incorrect output was less than one in a million, I will generate new test data and run it again. (Of course, if you are wrong about the probability being so low, the program will not get a second chance.) Furthermore, if you send me a pre-submission, and I report back to you that the output is correct for the test data, then it does not matter what the probability of incorrect output was. Still, I encourage you to aim for very low probabilities of incorrect output.

I will compile all programs using the g++ compiler on an Ubuntu 16.04 virtual machine that is allocated 2 GB of RAM, and I will test the executable under this environment; no compiler optimization options will be used for any program. I will specify the -std=c++11 flag when compiling, whether or not your code requires it. To be clear, I will compile the program like this: `g++ -std=c++11 <program_name>`. You should not send a Makefile (I will ignore it if you send one). You may only use libraries that are available with the standard g++ compiler. If you are unsure if a library is valid, I strongly recommend sending a pre-submission, so I can let you know if your program compiles and runs correctly.

Submit your program to me by e-mail. Send the code to *CarlSable.Cooper@gmail.com* as an attachment. Your program is due before midnight on the night of Wednesday, May 2. There is a lot of room for creativity with this assignment, so have fun with it!