Data Structures and Algorithms I
Spring 2019
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 three C++ string fields which store the last name, first name, and social security number (ssn) of a hypothetical person. The last names have been chosen randomly from a set of the 500 most common last names according to the 2010 U.S. census. The first names have been chosen randomly from a set of the 250 most common male names and the 250 most common female names in the U.S. between 2010 and 2017. (It shouldn't really matter if the data is accurate; I found it on-line, and I will post text files containing the last names and first names used for the assignment.) The ssns have been randomly generated and have the format ddd-dd-dddd, where each character 'd' represents a digit. (Within each dataset, ssns are guaranteed to be unique.)

After creating the list, the program will sort the list. Items should be sorted according to last names; if last names are identical, they should be sorted according to first names; if both last and first names are identical, they should be sorted according to ssns (which are guaranteed to be unique). 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, including a last name, first name, and ssn, 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').

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.

Your program will be tested on four test cases, which we will call $T_1 \ldots T_4$. For each of these test cases, a single file will exist using the format specified earlier; i.e., the first row will indicate the number of rows to follow, and each additional row will contain three strings representing a person's last name, first name, and ssn. The contents of the four test cases will also adhere to the following specifications:

- T_1 will contain approximately (within 1 percent of) 100,000 data objects. Each data object will store a randomly selected last name, a randomly selected first name, and a randomly generated social security number (ssns are guaranteed to be unique within T_1).
- T_2 will contain approximately (within 1 percent of) 1,000,000 data objects. Each data object will store a randomly selected last name, a randomly selected first name, and a randomly generated social security number (ssns are guaranteed to be unique within T_2).
- T_3 will contain the same data objects as T_2, but they have already been sorted according to last names and first names (but not according to social security numbers).
- T_4 will contain approximately (within 1 percent of) 1,000,000 data objects. Each data object will store the same last name and the same first name (both selected randomly, but only once), and a randomly generated social security number (ssns are guaranteed to be unique within T_4).

Data Structures and Algorithms I
Spring 2019
Programming Assignment #2

Every working program will be assigned a score that is based on the CPU times that the program takes to sort the four test cases. If $time_1 \ldots time_4$ are the CPU times required by the program when tested on T_1 through T_4, respectively, 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 test cases), *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 standard C++ 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 probably 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 (although not right away), 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.)

You may only submit one final program, which I will use for all four test cases. The program will not be told which test case it is dealing with; however, the program can try to figure out which test case it is dealing with, and it may use different strategies for each. You don't really have to get that perfect, and in general, your program does not have to work for all possible test cases, as long as it ultimately works on my actual test data. (I therefore highly encourage pre-submissions). 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.

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>`. Please do not use features that were introduced in version C++14 or later. You may only use libraries and functions that are available with the standard g++ compiler. If you are unsure if a library or function is valid, I strongly recommend sending a pre-submission, so I can let you know if your program compiles and runs correctly. You should not send a Makefile (I will ignore it if you send one).

Submit your program to me by e-mail to *CarlSable.Cooper@gmail.com* as an attachment. Only send your code (do not send object files, test data, or executable files). Your program is due before midnight on the night of Sunday, May 5. There will be a late penalty of 5 points per day, and I will not accept programs after Wednesday, May 8. I will start accepting pre-submissions on Saturday, April 27. There is a lot of room for creativity with this assignment, so have fun with it!