ENSC 251 Lab Assignment 3

In the ENSC 251 course, you will work on a multi-lab project: you will design and implement a simplified graduate student admission system using the skills learned throughout this course (Object-Oriented Programming with C++). This project will be developed incrementally throughout a total of four labs, each weighing 10% of your final grade marks. All lab assignments will be carried out and evaluated in pairs. Some general grading logistics have been posted on our course website: https://coursys.sfu.ca/2019fa-ensc-251-d1/pages/labs. Please note that the detailed grading scheme for each lab will be released after your lab grading though: think about you are in a real interview, nobody will tell you what the detailed grading schemes are.

More description of the simplified graduate admission system was given in lab assignment 1 and 2. Now we are moving to lab assignment 3, which is a further enhancement based on lab assignment 2. For those student groups who didn't do well in assignment 2, the TAs will post the code of assignment 2 from those top student groups who got bonus points, and you are free to use their assignment 2 code as basis for your assignment 3.

Lab Assignment 3:

Assume there are a number of domestic and international student applicants who are applying to SFU graduate school, and most of their profile is stored in domestic-stu.txt and international-stu.txt, which you are already able to read in assignment 1 and 2. In lab assignment 2, we assumed there are a fixed number of domestic and international applicants and used array/vector to store them and sort them. We continue to enhance this graduate admission system in lab assignment 3.

Now in lab assignment 3, we will use ***singly linked lists*** to store the DomesticStudent and InternationalStudent objects ***in a sorted order***, so that we can ***quickly insert a new applicant or delete an existing applicant***. Note that a binary search tree could be a better data structure than a singly linked list, but we would simplify the requirements in this assignment to make your life easier. You will have to achieve the following goals:

1. Use one singly linked list to store all the DomesticStudent objects in a sorted order and another singly linked list to store all the InternationalStudent objects in a sorted order, which are read and initialized from the input files.
    a. Whenever you read one line of data from the input file, initialize a DomesticStudent (and InternationalStudent) object, and insert this object into the DomesticStudent (and InternationalStudent) singly linked list. You have to modify your DomesticStudent (and InternationalStudent) class to make it a linked list node. Note the linked list node type should be a class instead of a struct in this assignment.
    b. Each time you insert a DomesticStudent (and InternationalStudent) object into the DomesticStudent (and InternationalStudent) singly linked list, you have to make sure that all objects in the singly linked list are sorted based on the overall sorting scheme that you implemented in lab assignment 2. That is, your singly linked list is always sorted.
    c. You should maintain both a ***head*** pointer and a ***tail*** pointer that points to the head node and tail node of your DomesticStudent (and InternationalStudent) singly linked list.

2. Based on the user input, your program should be able to
    a. Search existing DomesticStudent (and InternationalStudent) objects in the DomesticStudent (and InternationalStudent) linked list based on the user input information “application id”, or “cgpa”, or “researchScore”. Print out all objects which have the same application id, or cgpa, or researchScore, as the user input. If there is no match, print out information indicating there is no match found. Note each search here should just take one input (e.g., cgpa) and basically you should have three separate search functions for application id, cgpa, and researchScore.
    b. Search existing DomesticStudent (and InternationalStudent) objects in the DomesticStudent (and InternationalStudent) linked list based on the user input information “firstName and lastName”. Print out all objects which have the same firstName and lastName (both matched) as the user input. If there is no match, print out information indicating there is no match found.
    c. Create a new DomesticStudent (and InternationalStudent) node based on the user input information, and insert this new node into the DomesticStudent (and InternationalStudent) singly linked list in order (using the overall sorting scheme in lab assignment 2).
    d. Delete existing DomesticStudent (and InternationalStudent) objects in the DomesticStudent (and InternationalStudent) linked list based on the user input information “firstName and lastName”. Delete all objects which have the same firstName and lastName (both matched) as the user input.
    e. Delete both the head node and tail node from the DomesticStudent (and InternationalStudent) linked list in a single delete function.

3. A final step, based on the user input, merge the two sorted DomesticStudent and InternationalStudent linked lists into a single Student linked list, which is also sorted based on a modified overall sorting. And then print all the Student objects information. That is, you print out the information of all Student objects (except those InternationalStudent objects who didn’t meet the TOEFL score requirement), which are sorted based on their CGPA and researchScore.
    a. When you merge the two lists, the modified overall sorting is similar to the one used in lab assignment 2, but slightly different. First they are sorted based on their research score. If they have the same research score, then they are further sorted based on their CGPA. If they further have the same CGPA, then they are merged in their original order except that you put all DomesticStudent objects ahead of all InternationalStudent objects.
    b. Search existing Student objects in the merged linked list based on the user input information “cgpa_threshold and researchScore_threshold”. Print out all DomesticStudent and InternationalStudent object information who have both a cgpa >= cgpa_threshold and a researhScore >= researchScore_threshold; basically, these are the students who will be admitted to SFU. If there is no match, print out information indicating there is no match found.

4. **!!IMPORTANT NOTE!!** You have to write your own code for all the above tasks, no library function calls (e.g., sort, list/deque containers, etc.) are allowed.

In addition to the actual coding implementation, you need to provide good commenting, naming, and other good coding styles (including the header file style to avoid the multiple includes problem); all these count in your lab assignment marking.

Note: For grading logistics and remote machine access, please refer to the course website. If you have any questions, please post them on Piazza.

Assignment Submission:

Your lab assignment 3 will be submitted electronically through CourSys. You will need to submit a single lab3.zip file. Failure to comply with this format could result in zero score on this assignment.

Submission Deadline:

Your lab assignment 3 is due at **11:59:59pm on Monday, Nov 4th, 2019**. You need to meet the deadline: every 10 minutes late for submission, you lose 10% of this lab mark; that is, 100 minutes late, you will get zero for this lab.

Lab Demonstration:

You will have to demo your lab assignment 3 to your TA in the following lab sessions that you enrolled in: Tuesday (**Nov 5th, 2019**), Thursday (**Nov 7th, 2019**), and Friday (**Nov 8th, 2019**). Only code from your CourSys submission is allowed in the lab demo. Each student group has around 6-7 minutes to explain your code to the TA. If you fail to do the demo (without a medical note), or if it is determined that you do not understand the code being evaluated, you will be awarded zero on this lab assignment. Also please show up in the demo day on time (beginning time of each lab session), otherwise you will lose 0.5 mark.