ENSC 251 Lab Assignment 1

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 (2 students per group) - I hope by now all of you have already found a partner. 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.

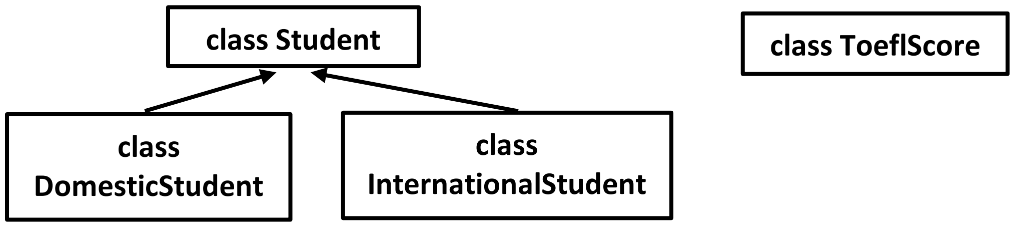
In this simplified graduate student admission system, we will have a number of domestic and international students applying to SFU graduate school, each student will be evaluated based on his/her CGPA, research score, and optional English language testing scores (if the applicant is an international student). Only the top applicants who meet a certain set of criteria will be admitted to SFU graduate school.

We make some simplifications here. First, for CGPA, we assume all applicants use SFU's 4.3 CGPA scale. Second, we simplify the research score to an integer number ranging from 0 to 100 (full score is 100). Note in reality, research score is based on a number of factors, including the applicant's research experience, publication, recommendation letters and etc. Third, we assume all international students need to provide their TOEFL scores for the English language test.

Are you excited? Let's get started.

Lab Assignment 1:

In this first lab assignment, you will define classes to be used throughout all four lab assignments. As shown in the figure below, you will have to design and implement four classes.



1. Parent class **Student**. Each student should have a first name and last name with ***string*** type, a CGPA (SFU 4.3 scale) with ***float*** type, and a research score (0 to 100) with ***int*** type. Basic error checking is required to ensure that each student has a valid CGPA, research score. In addition, each student should have an **8-digit unique** application id with ***int*** type, starting with 20200000. We assume there are less than 10,000 applicants in our system.
2. Child class **DomesticStudent** inherits from class **Student**. Each DomesticStudent has an additional field indicating which province he/she comes from and it has a ***string*** type. Imagine this could be used to evaluate if the applicant is eligible for certain BC-specific scholarships.

3. Child class **InternationalStudent** inherits class **Student**. Each InternationalStudent has two additional fields: 1) country indicating which country he/she comes from and it has a ***string*** type; 2) ToeflScore indicating his/her TOEFL scores.
4. class **ToeflScore** has five fields: 1) reading, listening, speaking, and writing, where each is an ***int*** score from 0 to 30. 2) total score that is a sum of all reading, listening, speaking, and writing. Basic error checking is required to ensure that the scores are valid.

For each class, you should design it to have proper constructor functions, get and set functions, make them abstract data types, and implement them in two files: a header file and an implementation file (i.e. student.hpp & student.cpp).

To test your classes, in the main function of main.cpp, you will need to 1) read input data from two files (domestic-stu.txt and international-stu.txt), and 2) use those read data to declare and initialize your class objects (in addition, you should create a unique application id for them), 3) manipulate these objects (e.g., set some fields of the DomesticStudent and InternationalStudent objects), and 4) print out the data members of these DomesticStudent and InternationalStudent objects to the screen. For the first step, I already provide you sample code (in main.cpp) to read data from an input file and process the data separated by a comma into each string, float, and int field. So, in this assignment, you can focus on the classes and objects.

To start the assignment, download the lab1-download.zip from the course website. It includes the following source code: **student.hpp, student.cpp, main.cpp, Makefile, domestic-stu.txt and international-stu.txt**.

1. After you download lab1-download.zip from the website and put it into a Linux machine
   a. [Optional] if you download it from a Windows machine, use **WinSCP** to transfer it to a Linux machine; you can download and install a WinSCP on your laptop
   b. unzip lab1-download.zip //decompress the zip file
   c. mv lab1-download lab1 //rename the directory to lab1
   d. You can see all files under the lab1 directory
2. Your focus should be modifying **student.hpp, student.cpp,** and **main.cpp**.
3. To compile and link your program, type "***make***". You should take a look at the gcc compilation and linking scripts though.
   a. If you develop in another computer other than the three virtual machines (i.e., ensc-esil-vm, vm2, and vm3), the compiler we use is gcc version 4.8.5 20150623. Keep in mind that your code should be able to compile, run, and demo on one of the three virtual machines though.
4. To run your program, type "***make run***".
5. To clean up your generated files, type "***make clean***".
6. For domestic-stu.txt, it includes initial datasets for domestic students. The first line is a description of the fields. For the rest of the lines, each line represents the data for one domestic student, including FirstName, LastName, Province, CGPA, and ResearchScore. Note that each field is separated by a comma and there is no white space.

7. Similarly, for international-stu.txt, it includes initial datasets for international students. The first line is a description of the fields. For the rest of the lines, each line represents the data for one international student, including FirstName, LastName, Country, CGPA, ResearchScore, and TOEFL Reading, Listening, Speaking, and Writing scores. Note that each field is separated by a comma and there is no white space.

In addition to the actual coding implementation, you need to provide good commenting, naming, and other good coding styles; 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 1 will be submitted electronically through CourSys. You will need to submit a single lab1.zip file. Failure to comply with this format will result in a **ZERO** score for this assignment. To zip your files in Linux,

1. Go to your lab1 directory
2. make clean //make sure you clean your files
3. cd .. //go one level up
4. zip -r lab1.zip lab //zip all your lab1 files into a single lab1.zip

Submission Deadline:

Your lab assignment 1 is due at **11:59:59pm on Wednesday, Sept 25th, 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 1 to your TA in the following lab sessions that you enrolled in: Thursday (**Sept 26th, 2019**), Friday (**Sept 27th, 2019**), and Tuesday (**Oct 1st, 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.