ENSC 251 Project

Project Overview:

In the ENSC 251 project: you will design and implement a simplified graduate student admission system using the skills learned throughout this course, i.e., Object-Oriented Programming with C++. Hopefully you can learn more about the programming skills, as well as how to get into a graduate program. The project weighs 50 marks, which includes 20 marks for the interim report and 30 marks for the final presentation. All students need to form a 4-person team to work on the project and will be evaluated per team (including both individual marks and team marks). Some general grading logistics have been discussed in Lecture 0 and posted on our course website: https://canvas.sfu.ca/courses/70216/pages/project-logistics.

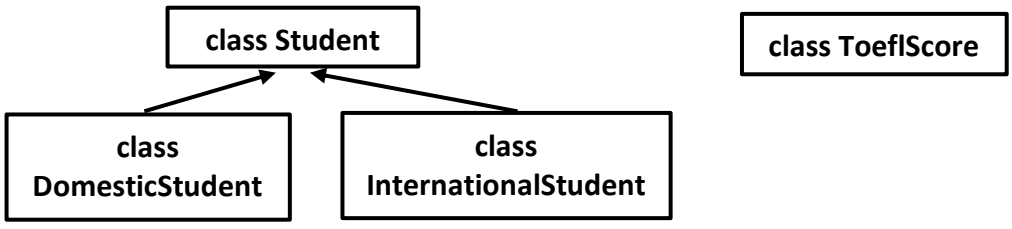
You can divide the work roughly based on the listed tasks; the overall marks for major tasks are also listed for your guidance, but not the detailed marks for every single item. Please include a simple document in the project report indicating which parts each member finished. There is some negative marking if you don't complete the listed tasks; details are not listed. Basically, the detailed grading scheme (except the overall marks for major tasks) for the project will NOT be released before your project grading is done. 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 satisfy certain criteria will be admitted to SFU graduate school.

We make some simplifications here. First, for CGPA, we assume all student applicants will use SFU 4.3 CGPA scale (round to single digit after the decimal point, e.g., 4.33 to 4.3, 3.67 to 3.7). Second, we simplify the research score to an integer number from 0 to 100 (full score 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.

Object Modeling:



As shown in the figure above, 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. In

addition, each student should have an **8-digit unique** application id with ***int*** type, starting with 20220000. We assume there are less than 10,000 applicants in our system.

2. Child class **DomesticStudent** inherits 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 scores.

Your Tasks for Interim Report (20 marks, due Jun 25th, 2022):

For the interim project report, you will have to implement the following tasks. At this moment, assume we have a fixed number of domestic and international applicants that we know beforehand, and both numbers don't exceed 100 (i.e., no more than 100 domestic student applicants, and no more than 100 international student applicants).

1. [4 marks] For each class, you should design proper constructor functions, get and set functions, make them abstract data types (ADT) and implement them in two files: the header file student.hpp and implementation file student.cpp. Also keep in mind about the class inheritance as described in the object overview.
2. [2 marks] Implement the following friend functions for Student class—**compareCGPA, compareResearchScore, compareFirstName, compareLastName**—so that you can compare the CGPA, research score, first name, last name of two Student objects. Each of these functions should take two Student objects (const Student& stu1, const Student& stu2) as function arguments, and return three kinds of values: less than, equal to, or greater than.
3. [1 mark] Overload the << operator of DomesticStudent and InternationalStudent classes to print out the object information.
4. [1 mark] Use arrays or vectors to store all the DomesticStudent and InternationalStudent objects, which are read and initialized from the input files (domestic-stu.txt and international-stu.txt). For this 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.
5. [4 marks] Implement sorting by each single field. Based on the user input, sort all DomesticStudent (and all InternationalStudent) objects in the array or vector by their CGPA, research score, first name, last name using these friend functions, and print out the sorted array or vector of objects. For CGPA and research score, sort them from high score to low score; and for first name and last name, sort them using ascending order, e.g., "Alex" comes before "Mary". Note each time the sorting is based on a single field**,** e.g., based on only CGPA, or only first name. Your program should be able to take input from a user (hint: using cin). For example, if user types in 'c', you should sort all DomesticStudent (and all InternationalStudent) objects based on their CGPA, and print out the sorted objects.

6. [4 marks] Implement the overall sorting by all fields. Based on the user input (if a user chooses an overall sorting), sort all DomesticStudent (and all InternationalStudent) objects in the array or vector, such that they are first 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 sorted by their province (for DomesticStudent) or country (for InternationalStudent) using ascending order. For InternationalStudent, if one's TOEFL score doesn't meet certain conditions, we drop that student object from the sorting process. The standard is as follows: the minimum overall TOEFL score is 93 with a minimum of 20 in each category (reading, listening, speaking, and writing). In contrast to point 5, where sorting is based on a single field, this overall sorting is based on multiple fields as we described. Note that you only drop those students in the output array/vector, but still maintain them in the input array. And you may want to add some friend function or operator overloading to the DomesticStudent and InternationalStudent class for the overall sorting. Similarly, after this overall sorting, you will print out all the sorted objects.
7. For all these sorting functions, each time, you only sort all DomesticStudent objects or all InternationalStudent objects; you don't need to sort between a DomesticStudent object and an InternationalStudent object. A better approach is to first ask a user to select DomesticStudent or InternationalStudent, and then ask a user to select the field(s) that they want to sort.
8. For all these sorting functions, they take an unsorted input array and give a sorted output array. They are implemented as normal functions outside classes. The function declaration and definition should be separated in stu_sort.hpp and stu_sort.cpp files. Moreover, the sorting performance is also important and plays a weight in the marking; please choose an efficient sorting algorithm.
9. Your program should print out a menu to the user for selecting corresponding operations as mentioned earlier. Before a user selects to quit the program, your program should keep running.
10. Make sure you add basic error checking for your code.
11. [3 marks] Make sure you test your program with a comprehensive coverage, including the regular cases, corner cases, and illegal cases.
12. **!!IMPORTANT NOTE!!** You have to write your own code for all the above tasks, no library function calls (e.g., sort API) are allowed. The vector class and string type are allowed.
13. Always note that in addition to the actual coding for implementation, you need to provide good commenting, naming, and other good coding styles.
14. [1 mark] Write a simple document (in .pdf format) about 1) task distribution between members; 2) testing plan (including the regular cases, corner cases, and illegal cases); and 3) weekly project schedule to plan and track your project progress. Include this pdf file in your final proj.zip file for submission.

In the project code you download from the course website, I already include the above files, including student.hpp, student.cpp, stu_sort.hpp, stu_sort.cpp, main.cpp, Makefile, domestic-stu.txt and international-stu.txt.

1. For domestic-stu.txt, it includes initial datasets for domestic students. The first line is a description of what each filed means. 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 filed is separated by a comma and there is no white space.

2. Similarly, for international-stu.txt, it includes initial datasets for international students. The first line is a description of what each filed means. 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 filed is separated by a comma and there is no white space.
3. Note there are some intentional typos (e.g., Idian) in the input files, which don't need to be fixed for now.
4. Feel free to add more files and update Makefile to make your code more modular.

Please make sure your code can compile and run correctly on the lab computer. If your code cannot compile, then your group can get at most 10 marks. If your code can compile, but cannot run, then your group can get at most 12 marks.

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

Interim Report Submission:

Your interim project report will be submitted electronically through Canvas. You will need to submit a single proj.zip file, including the pdf document in proj.zip. Failure to comply with this format could result in zero score on this assignment. To zip your files in Linux,

1. Go to your proj directory
2. make clean //make sure you clean your files
3. cd .. //go one level up
4. zip -r proj.zip proj //zip all your proj files (including report) into a single proj.zip

Interim Report Submission Deadline:

Your interim project report is due at **11:59:59pm on Saturday, Jun 25th, 2022**. You need to meet the deadline: every 10 minutes late submission, you lose 2 marks; that is, 100 minutes late, you will get zero on this interim project.

Interim Report Demonstration:

You will have to demo your interim project to your TA in the following lab sessions that you enrolled on **Monday (Jun 27th, 2022**). Only code from your Canvas submission is allowed in the lab demo. Each student group has around 16 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 interim project report. Also please show up in the demo day on time (TA will send out your scheduled time), otherwise you will lose 0.5 mark.