**Project #1 Report**

**Introduction:**

In this project, we had multiple classes related by being members of each other. We had a class Students that had the student’s name and personal information, and then a class Courses which had members representing homework, quiz, and test grades. A class StudentCourses connects these two by having members of both classes so that a student has personal information as well as grades for a course. A final class GroupOfStudents contains a member that was a vector of StudentCourses; its purpose is to be a list of students for a classroom, containing all students in a classroom (their information and grades).

* **Experience:** Working with these classes was different, since before, we usually worked with classes related by direct inheritance, and we had never encountered a class that is defined by two other classes. Also, when dealing with inheritance before, the parent classes were usually not used, and were only there to define general functions and members and be inherited from, like Shape. These classes, however, were independently used in different parts of the lab, such as separately reading in students and courses data or calculating final scores in Courses, as well as used together, such as when the StudentsCourses or GroupOfStudents objects were used to display sorted information. The StudentCourses was especially interesting in that it “inherited” from two classes instead of one; we hadn’t run into that before. This was a new concept to us, and we could see how it could lead to problems, such as conflicting definitions for functions. We also had to manipulate these classes based on specific members, such as sorting last names and searching for a student by their score. Now having experience with this, we feel that we have a lot more flexibility in using custom classes.
* **Learning Outcome:** After this project, we feel that we have a better handle on making extensive custom classes, relating classes on more than just inheritance, and using related classes together as well as alone. We also gained more experience in using functions on classes (like sorting and searching) that focus on specific members. In the future, we should be able to use classes more naturally and navigate through them easier and without confusion.
* **Insight:** Group works is an essential part of real life programming project. By doing this assignment we have a bit more sense in group collaboration. We learned to write codes and comments such that the other person could read, understand and trust each other in a sense that one can confidently incorporate that code knowing that the code will work. So that we can each work on multiple tasks individually without hindering the progress of the project.

**Instructions:**

**To build the program, these files must be present:**

- Courses.cpp

* Courses.h
* GroupOfStudents.cpp
* GroupOfStudent.h
* Menu.cpp
* Menu.h
* Student.cpp
* Student.h
* StudentCourses.cpp
* StudentCourses.h
* pa1main.cpp
* “Student data text file” follow this format:

+ *lastname firstname ID*

*quiz1 quiz2 quiz3 quiz4 quiz5 quiz6 quiz7 quiz8 quiz9 quiz10*

*homework1 homework2 homework3 homwork4 homework5 homework6*

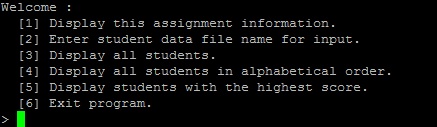
*test1 test2 test3 test 4*

**To run the program**, locate the program in the correct directory. Make sure all files are within one folder:

**Compile** using this command: g++-4.7 –std=c++11 \*.cpp

**Run** using this command: ./a.out

**Input and output format:**

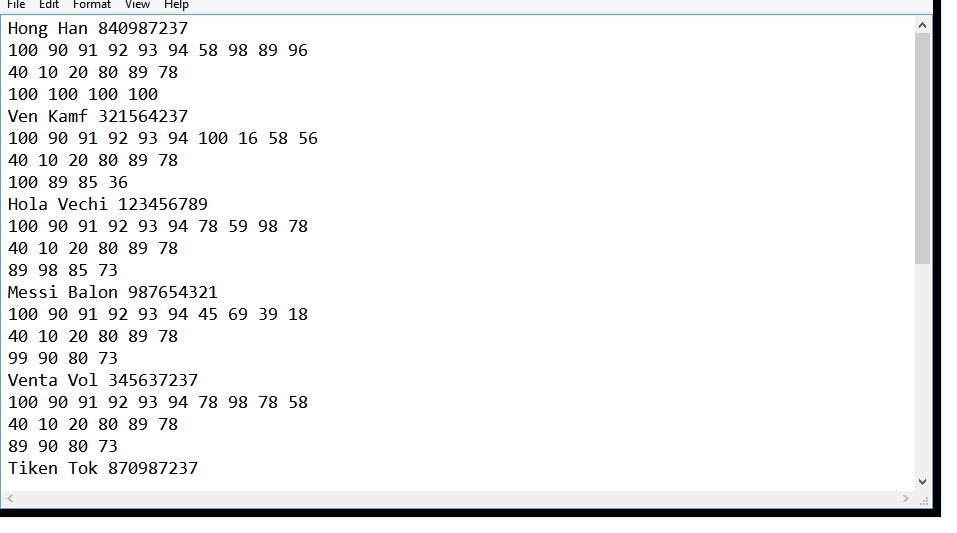
**Program option:**

**Data format:** *lastname firstname ID*

*quiz1 quiz2 quiz3 quiz4 quiz5 quiz6 quiz7 quiz8 quiz9 quiz10*

*homework1 homework2 homework3 homwork4 homework5 homework6*

*test1 test2 test3 test 4*

**** *Sample:*

**Exceptions:**

* **File doesn’t exist**: The program is able to detect and throw error if the file doesn’t exist.
* **Catch if a grade is not a digit**
* **Catch if a grade is out of bound (< 0 or > 100)**
* **Invalid data (ID contain letters, etc.):** The program is also able to detect and display the incorrect information that is needed to be fixed

**Algorithm:**

Bubble sort (Menu::display\_student\_sorted()):

* In the bubble sort, a vector of StudentCourses is inputed to be sorted by their last name, so that the students can be displayed alphabetically.
* In the inner loop, the sort goes down the list and performs “checks”; it compares the current element of the vector (the name of the current student) to the next one, and if the next one is smaller (closer to the beginning of the alphabet), it swaps their positions.
* The outer loop runs as long as the int ‘i’ is less than the size of the vector and increments when the inner loop is finished (when one round of checking is done, which means one element in the vector has been fully sorted). The int ‘i’ is then used to shorten the number of inner loop comparisons after every outer loop run. The inner loop runs as long as int j (which starts at 0) is less than the size of the vector – 1 (since there are one less side-by-side comparisons than there are elements being compared) – i. “i” is subtracted because after one iteration of the outer loop, one of the elements has been sorted as much as it needs to, so it would be unnecessary to perform another check with the finished element.
* Due to the two for loops, the worst-case complexity for bubble sort is O(n2) and on average, both loops will be used, so the average-case complexity is θ(n2). However, the best-case complexity is Ω(n) because the best case would be if the inner loop only had to iterate once.

**Classes:**

* Student:

+ Take student last name, first name, and ID.

* Courses:

+ Take grades of quizzes, home works and tests.

* StudentCourses:

+ Take in student and course class and group them together to become a unique collection of data.

* GroupOfStudent:

+ A collection of StudentCourses.

* Menu:

+ Implement the functions of the built classes into options for user interaction.

* InvalidFile:

+ Throw error, and customize the error output, can be deployed throughout lines of code.

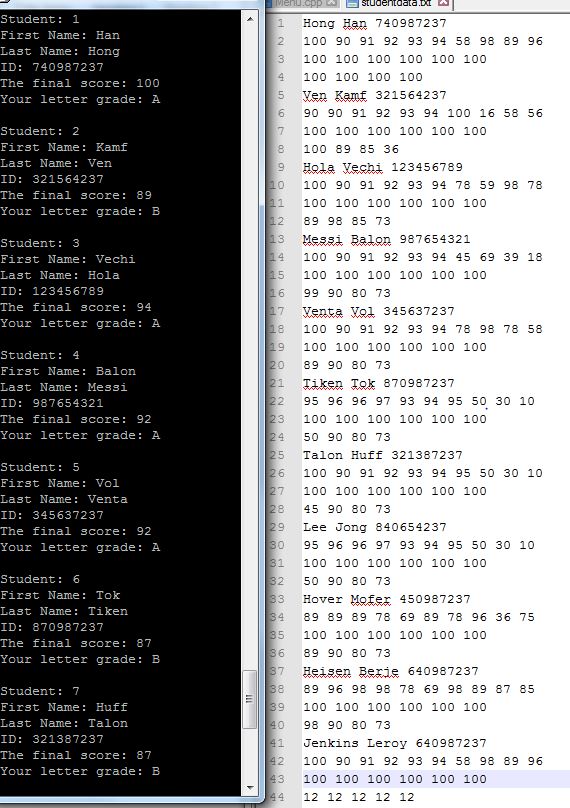
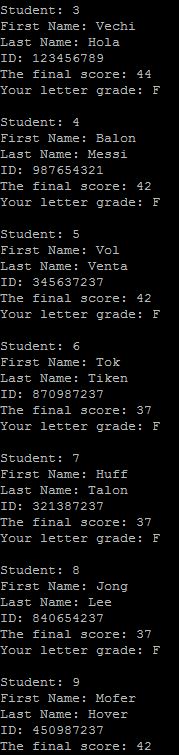
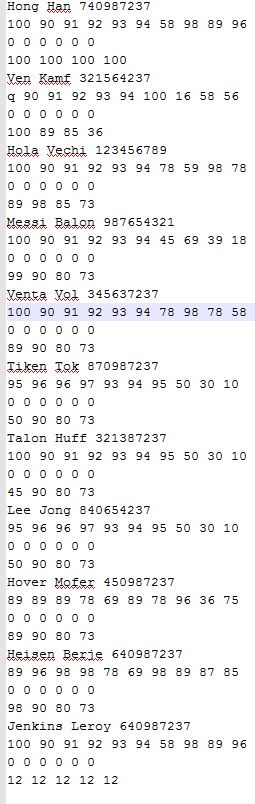
* InvalidData:

+ Eliminating the corrupting potential due to data input, like the InvalidFile, it can be deployed throughout the code where appropriate.

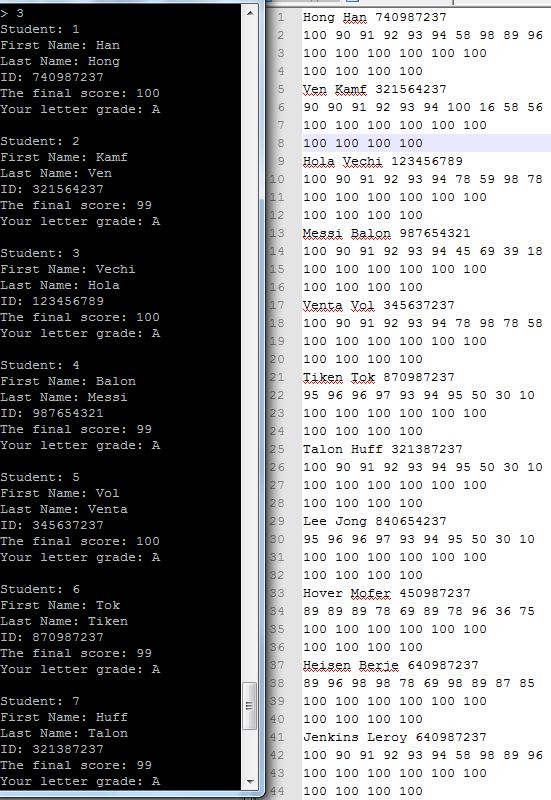
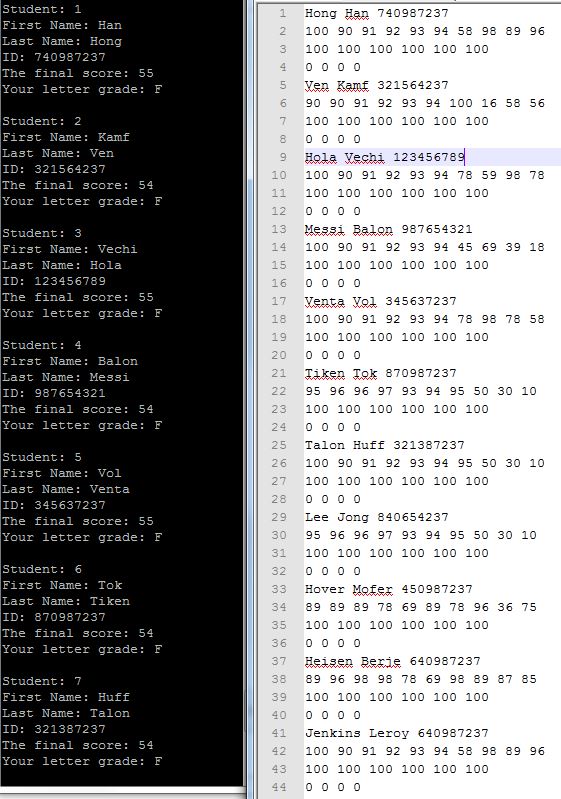
**Test cases:**

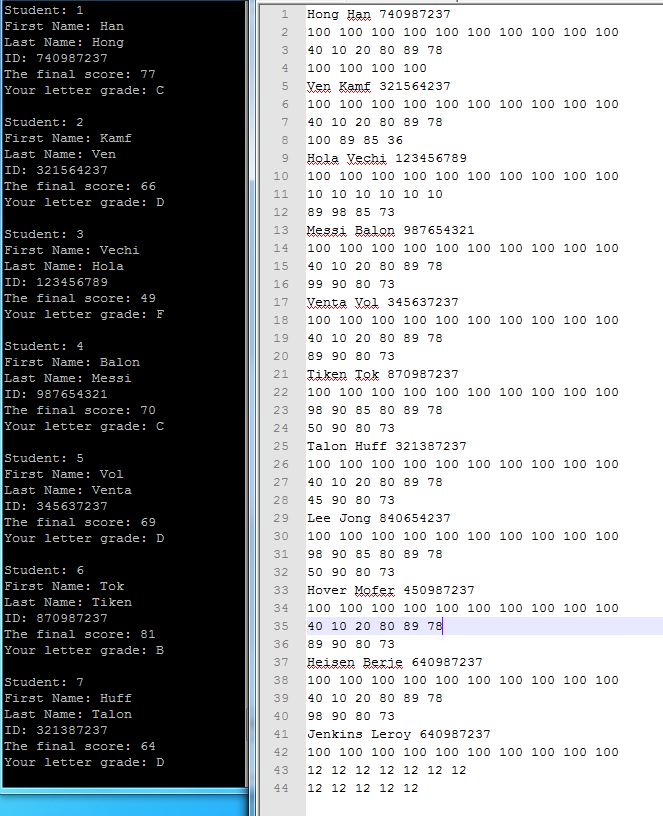
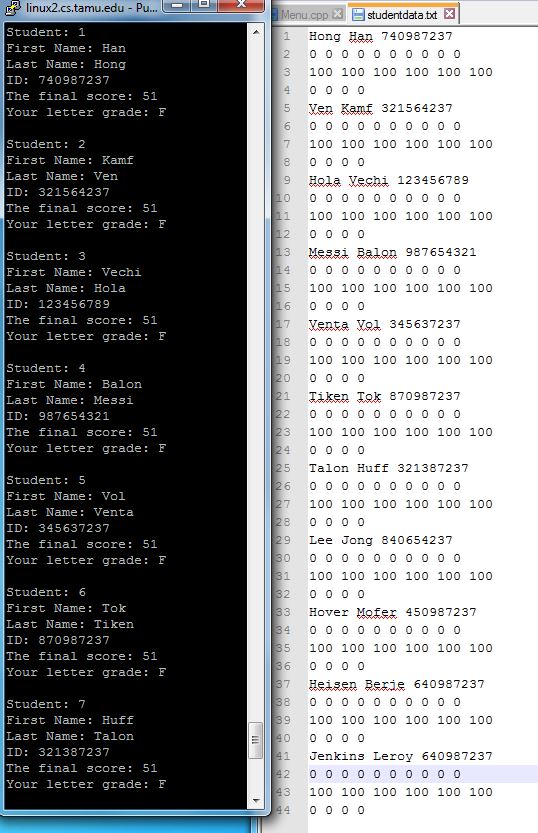
**Valid Case:**

1. **Quiz grades are 0: 2. Quiz grades are 100**



**3. Test grades are 0 4. Test grades are 100**

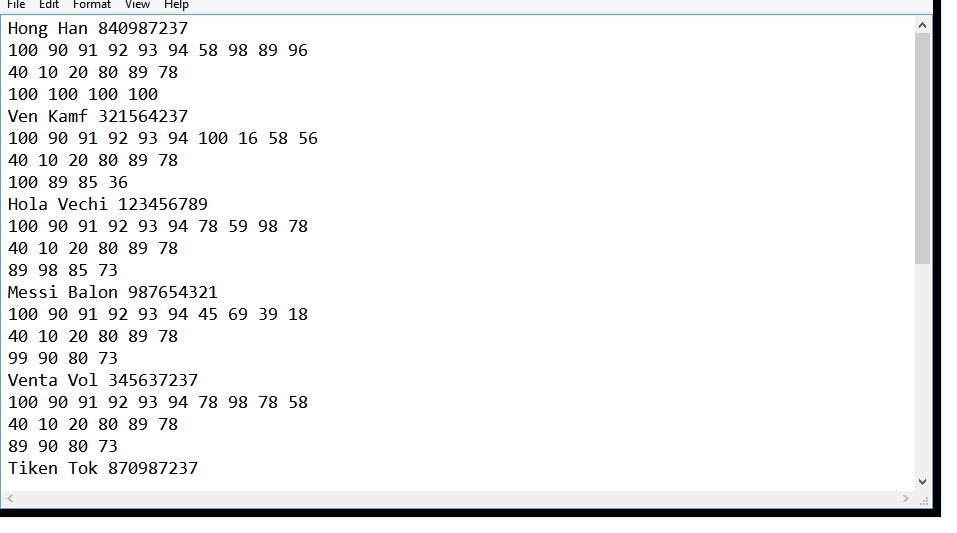
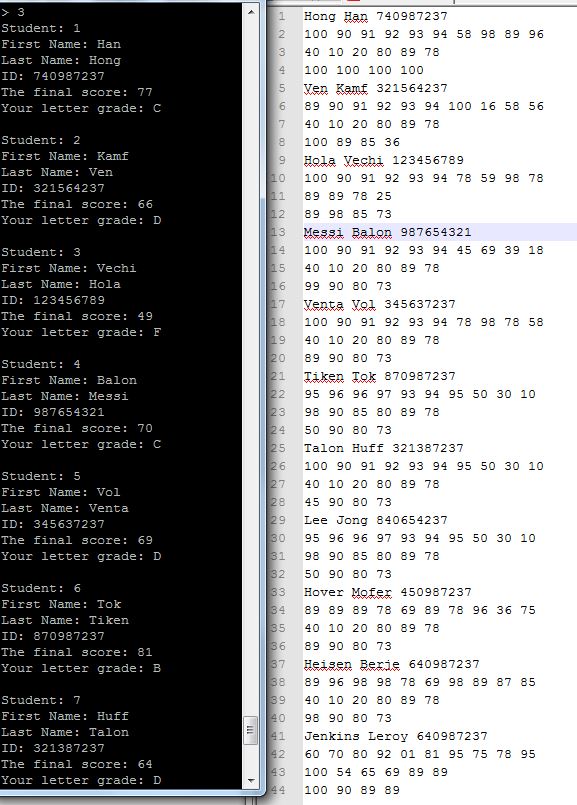
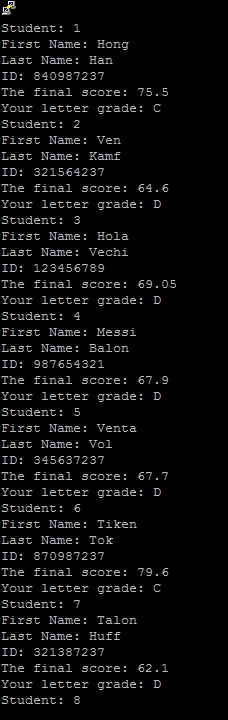


**5. Homework grades are 0 6. Homework grades are 100**

**Reason:** The purpose of these test cases (1 to 6) is to test the expected boundary of grades (homework, tests, quizzes), which is 0 -> 100. The program should run without any trouble when these values are in boundary, which is true in the above test cases.

**Random Cases:**

1. **2.**

****

**Reason:** Random test cases above demonstrate the program correct functionalities. The random test cases above run without error. Here is a hand calculation for one student (student Han Hong) in test cases 2. In addition, the data is rounded at the end to yield to most accurate data:

HW grades: 100 + 90 + 91 + 93 + 94 + 58 + 98 + 89 + 96 = 809/10 \* 0.05 = 4.045

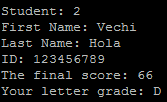
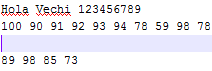
Quiz grades: 40 + 10 + 20 + 80 + 89 + 78 = 317/6 \* 0.5 = 26.41

Test grades: 100 + 100 + 100 + 100 = 400/4 \* 0.45 = 45

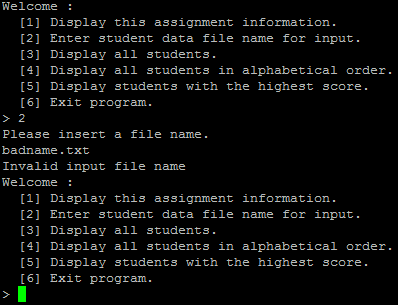
**Total: ~77 = C**

**Invalid Cases:**

**When a line of data is deleted, the code does not notice instead of throwing an error**



**When file doesn’t exist, the program correctly throw an error, alert the user that the file does not exist.**



**Reason:** These test cases are invalid, and is pressed again the program to check if the program behave correctly. For the empty, we found out that the program can’t catch it

**Known issues:**

* Can’t check if the name of the student contain numbers.
* Can’t catch empty lines.

**Potential Improvements:**

* A more throughout error checking using regex function could have solve most our format error. The method would be taking the string, compare to the regex pattern “.\*\\d.\*”. If it matches, that means the string does contain error, and therefore an error alert will be throw. Boost library will be the additional requirement to the program.
* Current data is displayed in a manner that is hard to read, we could have used align to put data in a table format, making it more eye candy.