# Assignment 6: Student Grades

Assigned: November 19, 2016
Due: December 02, 2016, 11:59:59pm

## Purpose

The purpose of this assignment is to provide you with experience in working with base and derived classes, virtual functions, and polymorphism. You will also gain more familiarity with file I/O.

## ABET/SMALCS Assessment

This assignment is designated as one of the course assignments being used to assess basic programming skills for ABET/SMALCs requirements. Please see the syllabus for details. Note that in addition to the normal grading scales, each student's submission will be judged in several aspects on a scale of “Highly Effective”, “Effective”, or “Ineffective”, as specified by ABSET/SMALCS outcome assessment procedures. A student's submission that earns 70% of the available points will count as an overall score of “Effective”.

## Project Task

You will design a set of classes for storing student information. You will also create a main program that will read student information from a file, store the data, compute final grades, and then print a summary report to an output file.

## Documentation

In addition to the project implementation, you will also need to provide a `README` file. Here you will need to document at least 5 errors you encountered and how you solved them. The errors can be some combination of compile errors, linker errors, runtime, and runtime errors. However, you can not use the same errors you used for the first two assignments. The format needs to be a numbered list where each item describes the error type, the error description, and the steps taken to fix the error. For the description, you can just describe the g++ message for compile/linker errors or the crash / wrong output for runtime errors.

You can also include documentation for places where your implementation is unfinished. If you document buggy or unfinished code and describe what the problem is and and general approach to fixing it, then point deductions will be less severe. This shows me that you are aware of problems in your code and that you at least have an idea of how to fix them. This is completely optional, however, it can only help you as I'm not going to change my test code based on your documentation. That is, if you report an error I would have otherwise missed, you won't lose any points for it.

## Part 1: Class Design

Design a set of classes that store student grade information. There should be one base class to store common data, and three derived classes that divide the set of students into three categories: English students, History students, and Math students. All data stored in these classes should be private or

protected. Any access to class data from outside should be done through public member functions. The base class should allocate storage for the following data (and only this data):

- Student's first name (you may assume 20 characters or less)
- Student's last name (you may assume 20 characters or less)
- Which course the student is in (English, History, or Math)

## Part 2: Calculate Average

Each class should have a function that will compute and return the student's final average, based on the stored grades. All grades are based on 100 point scale. Here are the grades that need storing for each subject, as well as a breakdown for computing each final grade:

- English: Attendance=10%, Project=30%, Midterm=30%, Final Exam=30%
- History: Term Paper=25%, Midterm=35%, Final Exam=40%
- Math: Quiz Average=15%, Test 1=25%, Test 2=25%, Final Exam=33%
    - There are 5 quizzes to be averaged into one Quiz Average (can be a decimal number).

## Part 3: Main Program

Has three stages:

1. Ask the user for input and output file names. This is the only input and output that should be done from keyboard and to the screen. All other input and output will be to and from files.
2. Read the student data from the input file and store it using an array of appropriate type. You should use just one array for all students, not a separate array for each subjects (i.e. a heterogeneous list). You will need to allocate this list dynamically since the size is stored in the input file. This also means that each list item will need to be created dynamically. Each student's data should be stored in a separate object. All dynamically allocated space should be cleaned up appropriately with delete when you are finished with it.
3. Print a summary report to the output file as specified below. You'll need to use the function that computes the final average when you do this since the final averages will be included in this summary report.

## Part 4: Input File Format

The first line of the input file will contain the number of students listed in the file. This will tell you how big a list you need. After the first lines, every set of two lines constitutes a student entry. The first line of a student entry is the name in the format `lastName, firstName`. Note that a name could include spaces, the comma will delineate last name from first name. The second line will contain the subject (“English”, “History”, or “Math), followed by a list of grades (all integers), all separated by spaces. The order of the grades for each class type is as follows:

- English: Attendance, Project, Midterm, Final Exam
- History: Term Paper, Midterm, Final Exam
- Math: Quiz 1, Quiz 2, Quiz 3, Quiz 4, Quiz 5, Test 1, Test 2, Final Exam

## Part 5: Output File Format

The output file that you print should list each students name as `firstName lastName` (no extra

punctuation between), Final Exam grade, final average, and letter grade. The final average is printed to 2 decimal places, and the letter grade is based on a 10 point scale (i.e. 90-100 A, 80-89 B, etc). Output should be separated by subject with an appropriate heading before each section, and each student's information listed on a separate line in an organized fashion. Data must line up appropriately in columns when multiple lines are printed in the file. At the bottom of the output file, print a grade distribution of all students (i.e. the number of As, Bs, Cs, etc).

## General Requirements

- No global variables other than constants
- All class member data must be private or protected
- Use the const qualifier on member functions where appropriate
- The code for this program should be portable. Test with g++ compiler on linprog before submitting
- You may use any of the standard I/O libraries that have been discussed in class (iostream, iomanip, fstream)
- You may use any of the C libraries that have been discussed in class (cstring cctype)
- You may use the string class library
- You may not use any of the other STL (besides string)
- Do not use any C++11 libraries or features

## Extra Credit

Within each subject in the output file list the students in alphabeteic order sorted by last name. Do not change the given case of the names that were read from the input file when you print the output file and do not change the output file format. Just print the records in order by last name. This sort needs to be true alphabetical and not just lexicographical sort.

## Submitting

- Remove all binary files and executables
- Use the tar utility to archive all of your class files, driver file, and `README` file
- Name the tar file according to the scheme: a5_<*last_name*>_<*first_name*>.tar
- Submit the tar file to the appropriate blackboard assignment link