Assignment 6: StudentGrades COP3330 Fall 2017

Due: Thursday, November 30, 2017 at 11:59 PM

Objective

This assignment will provide further experience with base and derived classes, virtual functions, and using applications of polymorphism. It will also provide practice 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. 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 the ABET/SMALCS outcome assessment procedures. A student's submission that earns 70% of the available points will count as an overall score score of Effective.

Task

You will design a set of classes for storing student information along with 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.

Program Details

1. 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 (assume 20 characters or less)
   - student's last name (assume 20 characters or less)
   - Which course the student is in (English, History, or Math)
2. 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 a 100 point scale. Here are the grades that need storing for each subject along with the breakdown for computing the final grade:
   - English
     * Term Paper = 25%
     * Midterm = 35%
     * Final Exam = 40%
   - History
     * Attendance = 10%
     * Project = 30%
     * Midterm = 30%

        * Final Exam = 30%
    - Math
        * Quiz Average = 15%
            - There are a total of 5 quizzes averaged together (result can be a decimal number)
        * Test 1 = 25%
        * Test 2 = 25%
        * Final Exam = 35%

3. Write a main program (in a separate file) that does the following (in order):
    - Ask the user for input and output file names. This is the only input and output that should be done from the keyboard and to the screen. All other input and output will be to and from files.
    - Read the student data from the input file and store it using an array of appropriate type. You should use just one array (i.e. a heterogeneous list) for all students, and not a separate array for each subject. You will need to allocate this list dynamically since the size is stored in the input file. Since the list is dynamic, the list items will also need to be created dynamically. Each student's data should be stored in a separate object, and dynamically allocated space should be cleaned up appropriately with delete when you are finished with it.
    - 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.

File Formats

Input File

The first line of the input file will contain the number of students listed in the file. This will test 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; that 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 – Term paper, Midterm, Final Exam
History – Attendance, project, Midterm, Final Exam
Math – Quiz 1, Quiz 2, Quiz 3, Quiz 4, Quiz 5, Test 1, Test 2, Final Exam

Output File

The output file that you print should list each student's name (`firstName lastName` – no extra punctuation between), Final Exam grade, final average (printed to 2 decimal places), and letter grade 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 (i.e. how many As, Bs, Cs, etc) of all students.

General Requirements

- No global variables other than constants
- All class member data must be private or protected
- Use the `const` qualifier on member functions wherever it is appropriate

- The code for this program should be portable. Test with g++ compiler before submitting
- You may use any of the standard I/O libraries that have been discussed in class (`iostream`, `iomanip`, `fstream`, `cstring`, `cctype`, etc) You may also use the `string` class library
- You may not use any of the other STL besides `string`
- Do not use any C++11 only libraries or features

Extra Credit

Within each subject in the output file, list the students in alphabetic order sorted by lastName. Do not change the given case (upper/lower 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

Archive your source files and `README` files into a simple tar ball (no compression). Submit to the assignment 6 link on blackboard. Make sure the submitted files are named as specified by the syllabus and this writeup, and that you do not include any extra files (e.g. object files or executables).

General Advice

- Make sure to double check your blackboard submission to make sure everything works when downloaded.
- Email a copy of your finished homework files to your own FSU account. This email will have a time stamp that shows when they were sent and will also serve as a backup. Useful in case something happens to blackboard.
- Periodically (e.g. nightly) make a backup of your assignment to another machine (e.g. personal computer, linprog, email). Computers die and accidents happen, having a backup prevents you from having to start from scratch. This is also a good feature to have in your makefile as you can abstract the details behind a single call.
- Make sure to include the `README` file as specified in the assignment syllabus http://ww2.cs.fsu.edu/~dennis/teaching/2017_fall_cop3330/docs/syllabus.pdf

COP 3330 Fall 2017 Assignment 6 Due: 2017-11-30 11:59 PM

Student Name: **Grader:**

Grade: **Date Submitted:** **Date Graded:**

Numeric Grade

Description	**Earned**	**Possible**	**Comments**
1. Private Data		2	
2. const Usage		3	
3. Memory Management		10	
4. Class Structure		10	
5. Class Relationships		10	
6. Object Array		10	
7. Terminal I/O		5	
8. File I/O		10	
9. Table Formatting		5	
10. Separated by Category		5	
11. Exam Calculation		5	
12. Summary		5	
13. Sorted		+5	
14. README		10	
15. Submission		10	

ABET Scores

Description	**Effectiveness**	**Comments**
Storage Facilities		
Classes		
Functions		
Control Structures		
Terminal I/O		
Code Quality		
Overall		

Notes: